Characterizing Neural Manifolds' Properties and Curvatures using Normalizing Flows

Peter Bouss,[1,2,*,†] Sandra Nestler,[1,2,3,*] Kirsten Fischer,[1,2] Claudia Merger,[1,4,5] Lars Schutzeichel,[1,2,6] Simon Musall,[6,7,8] Alexandre René,[1,9,10] and Moritz Helias[1,4]

[1]*Institute for Advanced Simulation (IAS-6), Jülich Research Centre, Jülich, Germany*
[2]*RWTH Aachen University, Aachen, Germany*
[3]*Rappaport Faculty of Medicine and Network Biology Research Laboratory, Technion – Israel Institute of Technology, Haifa, Israel*
[4]*Department of Physics, Faculty 1, RWTH Aachen University, Aachen, Germany*
[5]*International School of Advanced Studies (SISSA), Trieste, Italy*
[6]*Institute of Biological Information Processing (IBI-3), Jülich Research Centre, Jülich, Germany*
[7]*Institute of Biology 2, RWTH Aachen University, Aachen, Germany*
[8]*Faculty of Medicine, Institute of Experimental Epileptology and Cognition Research, University of Bonn, Bonn, Germany*
[9]*Department of Physics, University of Ottawa, Ottawa, Canada*
[10]*Faculty of Computer Science, RWTH Aachen University, Aachen, Germany*

(Received 20 June 2025; accepted 19 March 2026; published 27 April 2026)

Neuronal activity is found to lie on low-dimensional manifolds embedded within the high-dimensional neuron space. Variants of principal component analysis are frequently employed to assess these manifolds. These methods are, however, limited by assuming a Gaussian data distribution and a flat manifold. In this study, we introduce a method designed to satisfy three core objectives: (1) extract coordinated activity across neurons, described either statistically as correlations or geometrically as manifolds; (2) identify a small number of latent variables capturing these structures; and (3) offer an analytical and interpretable framework characterizing statistical properties by a characteristic function and describing manifold geometry through a collection of charts. To this end, we employ normalizing flows (NFs), which learn an underlying probability distribution of data by an invertible mapping between data and latent space. Their simplicity and ability to compute exact likelihoods distinguish them from other generative networks. We adjust the NF's training objective to distinguish between relevant (in manifold) and noise dimensions (out of manifold). Additionally, we find that different behavioral states align with the components of the latent Gaussian mixture model, enabling their treatment as distinct curved manifolds. Subsequently, we approximate the network for each mixture component with a quadratic mapping, allowing us to characterize both neural manifold curvature and non-Gaussian correlations among recording channels. Applying the method to recordings in macaque visual cortex, we demonstrate that state-dependent manifolds are curved and exhibit complex statistical dependencies. Our approach thus enables an expressive description of neural population activity, uncovering nonlinear interactions among groups of neurons.

DOI: 10.1103/rmfp-b7b5

I. INTRODUCTION

Understanding the neural code used by the brain to represent information is a central quest in neuroscience [1–10]. The question of representation encompasses two distinct aspects. The first concerns the relations between the activities of multiple cells. These relations can either be described *statistically*, for example in terms of correlations among the activities of multiple cells, or *geometrically*, in terms of the subspaces spanned by the joint activity of the entire neuronal population, also termed neural manifolds.

The second aspect of the neural code concerns the relation of the neuronal activity to informative lower-dimensional latent variables [11,12]. These latent variables can for instance represent movement parameters [13–16], encode sensory stimuli [17,18] or decision-making processes [19–21], or reflect other continuous internal states [22,23]. Inferring low-dimensional latent structure from high-dimensional neuronal activity is thus a key objective of computational neuroscience. Deciphering the neural code amounts to finding both, a description of the statistical or geometrical relations among the activities within a neural population, as well as identifying how latent causes are linked to this observed activity.

A recent line of research studies trajectories of neural activity in the space spanned by the activity of all neurons [24–32] and has found that these trajectories indeed do not fill the entire space but rather lie on manifolds with a lower intrinsic dimensionality than that of the embedding space [11,27,29,33]. Such manifolds imply that neurons are not activated independently of each other but must rather obey intricate dependencies. Gaining insight into the composition,

*These authors contributed equally to this work.
†Contact author: p.bouss@fz-juelich.de

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

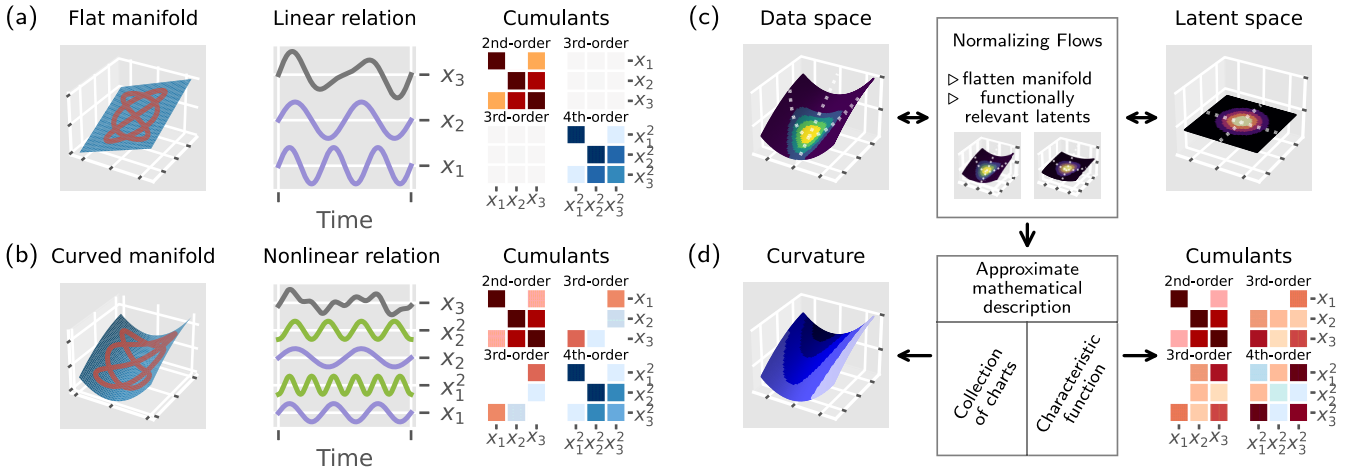


FIG. 1. Analysis pipeline for curved neural manifolds parameterized in terms of meaningful latent variables. (a) A conceptual signal trajectory x_i (red) embedded into a three-dimensional state space (left panel) but lying on a flat manifold (blue). Each axis represents one data channel. The middle panel displays the time evolution of each channel. The first two signals x_1 and x_2 (in purple) are independent oscillatory signals, while x_3 (in gray) is a weighted sum of x_1 and x_2. The relation is thus linear. In the right panel, cumulants of second, third, and fourth order are shown for combinations of x_i or x_i^2 and x_j or x_j^2. The vanishing of third-order cumulants indicates the absence of cross-channel nonlinear mixing, while the observed fourth-order cumulants here reflect non-Gaussian statistics of x_1 and x_2 rather than geometric curvature. (b) Left panel similar to (a): a trajectory (red) of a signal x_i lies on a curved manifold (blue). The construction is shown in the middle panel: The signal x_3 (gray) depends here on x_1 and x_2 (purple) as well as on their quadratic terms x_1^2 and x_2^2 (green). The manifold is curved due to this nonlinear relation. In contrast to (a), nonzero mixed third- order cumulants arise from nonlinear (quadratic) dependencies and therefore can indicate manifold curvature. (c) Left panel shows a probability density lying on a curved manifold. The normalizing flow (middle panel) learns to map both the distribution and its underlying manifold to a latent distribution on a flat manifold (right panel). In this deep neural network, every layer implements an invertible transformation. The latent distribution, described as a mixture of Gaussians, lies in a space whose axes (white dashed lines) correspond to the learned curved axes as shown in the original distribution (left panel); these are functionally relevant latent axes. (d) The trained normalizing flow is approximated by a quadratic expression, which can be used to create a collection of charts and a characteristic function (middle panel). The charts can be used to compute the curvature of the manifold (left panel). The characteristic function yields cumulants of different orders (right panel).

structure, and properties of these manifolds is central to understand how the brain represents and processes information. Numerous studies have explored neural manifolds for different brain areas, such as motor cortex [13,14], prefrontal cortex [17,34,35], hippocampus [36,37], visual cortex [9,18], and basal ganglia [24,25].

Principal component analysis (PCA) [38,39] is the most prominent method used in these studies [40], along with its many variants [13,35,41,42] including tensor-based methods [43,44].

PCA and related methods have been key in identifying low-dimensional structure, but their assumption of Gaussianity of the data distribution and the restriction to linear operations limit their ability to describe curved manifolds or complex dependencies in the neural activity. As shown in Fig. 1(a), a linear description suffices when the data lie on a flat manifold, for example, when x_3 is a linear combination of x_1 and x_2. In this case, there are exactly two linearly independent directions with nonzero variance, spanning a two-dimensional (2D) data manifold; in the absence of noise, this is exactly the 2D manifold identified by PCA, even when the distribution within the manifold is non-Gaussian. However, because PCA models data as lying on a linear subspace, any deviation from this subspace is treated as residual variance (often interpreted as Gaussian noise), making it unable to distinguish between noise and intrinsic manifold curvature. By learning a nonlinear mapping from the data to the latent space [Fig 1(c)], we are able to characterize the underlying structure more accurately.

With both non-Gaussian distributions and curved manifolds contributing to higher-order cumulants, these features need to be analyzed in a joint manner to characterize the data. In Appendix I, we show how to separate these contributions to the cumulants.

The assumptions underlying PCA, however, rarely hold for neural recordings from animals [31,45,46]: These often exhibit non-Gaussian statistics of the neural activity and curvature in the neural manifold—characteristics necessitating nonlinear methods for effective description and analysis. In Fig. 1(b), we illustrate such a curved manifold: Here the variable x_3 depends quadratically on x_1 and x_2, introducing curvature in state space and generating nonzero higher-order cumulants among the variables.

Various nonlinear methods have been proposed to characterize neural data embedded in curved manifolds, but they rarely provide a corresponding generative model.

Low-dimensional visualization tools such as diffusion maps [47–49], CEBRA [37], t-SNE [50], and UMAP [51] can represent such data effectively but do not offer a generative description or explicit modeling of the manifold structure. In contrast, classical manifold learning techniques such as ISOMAP [52], locally linear embedding [53], or Laplacian eigenmaps [54] provide a bijective map to a low-dimensional

latent representation, but they lack a probabilistic formulation that would enable data sampling.

Methods which simultaneously infer latent dynamics and geometry [46,55] can extract interpretable latent variables from neural populations, while approaches based on differential geometry [56] and topological data analysis [57] offer explicit descriptions of the manifold structure, however, without giving detailed insights into the statistical features underlying the data.

Latent dynamics can also be captured using deep generative models, such as sequential and variational autoencoders [22,58–67], but the resulting latent variables are difficult to interpret. To address this, a growing line of work has sought to adapt normalizing flows to manifold-supported data [68–78]: This in addition imposes a probability distribution on the latent space, thus improving their interpretability. Despite this advance, the complexity of latent dynamics still makes it challenging to relate latent variables to behavior, and the mapping from latent space to data space is not used for explicit characterization of the manifold geometry.

Methods that describe data lying on a lower-dimensional manifold are also vulnerable to overfitting due to degenerate probability distributions [64,66,72,73,77]. Hybrid approaches combine generative modeling with geometric structure [67,74,75,79], but the connection between latent dynamics, behavior, and manifold geometry remains challenging.

While these methods describe certain aspects of curved manifolds well, many share common shortcomings: They are difficult to interpret in terms of latent variables that describe animal behavior, or they lack an explicit description of the manifold's underlying curved geometry.

Our goal with this work is to address these shortcomings: We introduce a method that captures both the geometric structure and the resulting statistical dependencies within the data, while also offering a low-dimensional space of latent variables. Both geometry and statistics are extracted from the same analytical approximation of the mapping between latent and data space.

Concretely, in this work we introduce a method which simultaneously (1) learns a low-dimensional latent representation, (2) characterizes the data manifold's geometry, and (3) derives an interpretable description of the manifold's curvature and yields high-order statistics of the data. To the best of our knowledge, this is the first method to achieve this combination. The proposed method opens the door to a richer understanding of how neuronal activity is structured, beyond what current generative or dimensionality reduction techniques can provide.

Accordingly, we train invertible neuronal networks, also known as normalizing flows (NFs) [80–82], to learn a bijective mapping from the data space to a latent space; the training is such that this latent space is flat and the latent variables follow a simple distribution. Although there have been proposals to learn manifolds with normalizing flows [66,69–72,74–78], these do not address the three objectives that we aim for in this study. With our approach, we propose three key extensions to regular NFs: The first two are concerned with the design of the loss function and the latent space, and the third improves the interpretability of the learned manifold.

First, we enforce a low-dimensional structure in the latent space by a modified loss function [83], ensuring that the most relevant directions in latent space correspond to meaningful neural population dynamics. Mapping backwards from the latent to the data space, we thus obtain a curved, low-dimensional manifold in the data space that captures the most prominent features of the data. This concept is sketched in Fig. 1(c), where the axes of the latent space are shown as white dotted lines; when they are mapped backwards through the network, they serve as informative, curved axes describing the original data distribution.

Second, we replace the standard Gaussian latent distribution with a mixture of Gaussians. This allows the network to represent multimodal neural data—an essential feature for capturing state-dependent activity.

Third, to enhance interpretability, we approximate the learned network mapping with a quadratic function. We then obtain the correlations between neuronal activations by extracting an approximate characteristic function from the network mapping [see Fig. 1(d), right]. This approximation also allows us to obtain a set of differential geometry charts which in turn yield the Riemannian curvature tensor of the neural manifold [see Fig. 1(d), left].

To showcase the utility of our approach, we apply it to neural recordings from the macaque visual cortex [84]. In this data, we find different components, where some of them clearly align to the behavioral states. The other components show clear non-Gaussian higher-order correlations, and geometrically, these components lie on manifolds that are saddle-point like. These components are not so closely tied to one of the behavioral states. The components which are stronger aligned to the behavioral states show deviations from Gaussianity to a lesser degree and lie on less curved manifolds.

To demonstrate the broader applicability of our framework across recording modalities, we additionally apply our method to Neuropixels spike train recordings from mouse visual cortex and superior colliculus (Appendix D). This second dataset exhibits qualitatively different manifold characteristics, especially elliptic rather than saddle-point-like geometry. It shows significant non-Gaussian correlations and stimulus-dependent component structure. The different geometries between the two datasets highlight how our approach can reveal dataset-specific neural coding principles.

The remainder of the work is organized as follows:

Section II introduces the macaque visual cortex dataset and highlights the deviations from Gaussianity, with a focus on higher-order correlations. In Sec. III, we provide a concise overview into normalizing flows. Section IV outlines our additional reconstruction error loss function and illustrates, through a synthetic example, how it reveals a curved manifold in data. We then introduce a multimodal latent space in Sec. V, explaining how this enables the analysis of datasets with different behavioral states. Section VI details the selection of an appropriate multimodal latent space for our macaque visual cortex data.

We evaluate the alignments of the components that our model gives to the ground-truth behavioral states of the data in Sec. VII. Next, in Sec. VIII, we introduce a quadratic function as an approximation of the inverse map of the network,

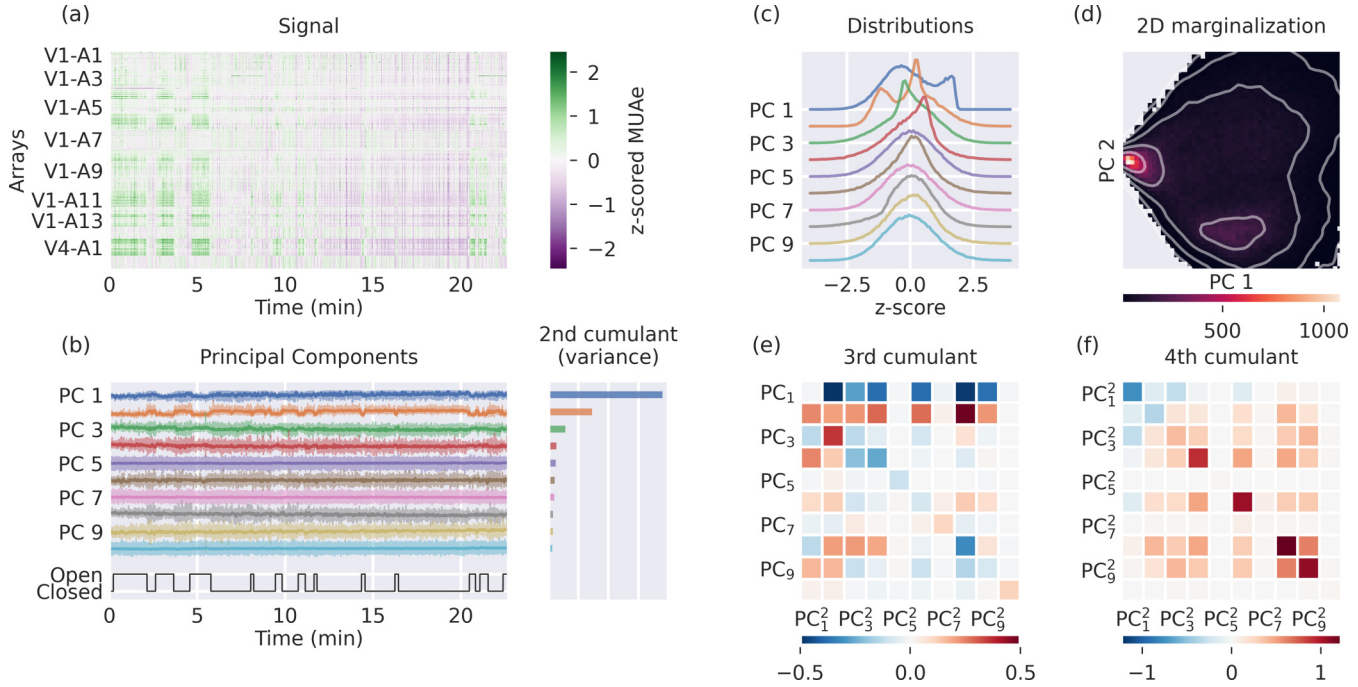


FIG. 2. Neuronal data exhibit higher-order correlations. (a) The original signal from one recording session (L_RS_250717), sampled at 100 Hz over 22.7 min. The signal is a MUAe containing 774 channels from 16 recording electrode arrays (14 in V1 and 2 in V4). Each channel is independently z scored (zero mean, unit variance) to ensure consistent scaling across neurons. (b) Time evolution of the first 10 PCs of the z-scored data. Transparent curves represent raw PCs, while solid curves show temporal smoothing using a Gaussian kernel with 1-s width. The labels of the behavioral state (eyes open or eyes closed) are shown below. The rapid decay of the eigenvalues σ_i^2 of the covariance matrix (second-order cumulants; on the right), corresponding to the shown PCs, is an indicator that the neural data lie on a lower-dimensional manifold. (c) Marginal distribution of the first 10 PCs. While some appear Gaussian, others clearly deviate from Gaussianity. (d) Two-dimensional marginalization along the first and second PCs. The color bar indicates the density of data points in the 2D histogram. [(e) and (f)] Third- and fourth-order cumulant. For both cases, the signals along the PCs are first divided by their standard deviation σ_i to normalize for the amplitude. Thus, panel (e) portrays x_i/σ_i against x_j^2/σ_j^2 and panel (f) x_i^2/σ_i^2 against x_j^2/σ_j^2. Color bars indicate cumulant magnitudes.

forming the basis for further analytical insight. Section IX leverages this approximation to compute the moment- and cumulant-generating functions for each of the components and analyzes higher-order statistical structure for each of them. The geometric properties of the inferred manifolds are explored in Sec. X, where we identify saddle-point-like curvatures.

Last, Sec. XI situates our approach within the broader literature, compares it to related models, discusses limitations, and outlines promising directions for future work.

II. CORRELATION AND GEOMETRY EFFECTS IN THE NEURAL CODE

To investigate the representation of information in cortical neuronal networks, we analyze electrophysiological recordings with hundreds of channels, each capturing multineuronal activity. Our primary analysis uses recordings from macaque visual cortices V1 and V4 obtained by Chen *et al.* [84], with comprehensive data details and preprocessing described in Appendix A. Additionally, we validate our approach using Neuropixels data from mouse cortex to demonstrate cross-species generalizability, shown in Appendix D. We will demonstrate in both datasets that neuronal activity significantly departs from a Gaussian distribution.

The macaque V1 and V4 recordings come from resting-state sessions where monkeys' eye states (open or closed) were monitored without task performance. Morales-Gregorio *et al.* [32] find that these two states correspond to two distinct neural manifolds using a Gaussian mixture model (GMM). We now examine these two manifolds in more detail.

We showcase this dataset and its properties for an example recording session in Fig. 2. The data consist of 774 signal channels. For each channel, the multiunit activity envelope (MUAe) is z scored over time. In Fig. 2(b), the time evolution of the 10 PCA variables (PCs) with the largest eigenvalues are displayed alongside the label of the eye state. We observe that certain PCs show a distinguishable behavior for the two different eye states. Furthermore, the eigenvalues of the PCs decay rapidly, indicating that a few variables explain most of the variance and thus supporting the hypothesis that the data lie on a low-dimensional manifold.

Inspecting Fig. 2(a) more closely, we note that the distributions for some channels appear to have heavy tails (having in mind that all channels are normalized to have unit variance). This suggests that certain channels deviate clearly from Gaussianity, a point we will quantify more systematically below using fourth-order cumulants.

The marginalized distributions in Fig. 2(c) indicate that the data are neither Gaussian nor a mixture of two Gaussians.

While for some PCs, the marginalized distributions are close to a Gaussian shape (e.g., PCs 5, 7, and 10), others deviate clearly from a Gaussian distribution (e.g., the first four PCs). The 2D marginalization of the probability density along the first two PCs in Fig. 2(d) appears to be bimodal, which is consistent with the ability of Morales-Gregorio *et al.* [32] to fit the data with a GMM of two components. However, the overall shape of the distribution differs markedly from what would be expected for a mixture of Gaussians: Instead of near-elliptical contours, the probability density is concentrated within a well-defined triangular region, suggesting strong deviations from Gaussian.

These visualizations already suggest non-Gaussian statistics. A more quantitative way to describe such deviations is through higher-order cumulants, which also capture statistical dependencies in high-dimensional spaces. For any Gaussian distribution, cumulants beyond second order vanish; nonzero values of such cumulants can thus be used as an indicator for deviations from Gaussianity. For this dataset, we observe significant third- and fourth-order cumulants [Figs. 2(e) and 2(f)]. While higher-order cumulants vanish for the close-to-Gaussian PCs 5, 7, and 10, they yield clearly negative (blue) or positive (red) values for other PCs, in particular PCs 1 and 2. The strongly positive values on the diagonal of the fourth-order cumulant may indicate heavy tails in these directions.

The analyses above confirm that the data exhibit structured, non-Gaussian statistics. As we pointed out in the Introduction, such higher-order correlations are closely tied to the curvature of the underlying manifolds. This connection highlights the need for a method capable of capturing these structures in order to gain a more complete understanding of the neural representation.

III. UNSUPERVISED LEARNING OF DATA DISTRIBUTION WITH NORMALIZING FLOWS

In order to learn the underlying distribution of the neuronal activity, we now employ NFs [80–82]—a class of invertible neural networks designed to learn complex, high-dimensional probability distributions that is used in different fields of machine learning. In the following paragraphs, we will introduce NFs and our adaptations which enable them to simultaneously learn a low-dimensional latent representation, characterize the geometry of the data manifold, and provide interpretable access to its curvature and higher-order correlations.

These neural networks define a bijective map

$$z = f_\theta(x) \tag{1}$$

from input data $x \in \mathbb{R}^N$ to latent variables $z \in \mathbb{R}^N$. The inverse

$$x = f_\theta^{-1}(z) \tag{2}$$

maps back from the latent space to the data space. The network parameters θ are learned so that the data distribution $p_x(x)$ is mapped to a simple distribution $p_z(z)$ in the latent space. An estimator $\hat{p}_{x;\theta}(x)$ for the true underlying distribution $p_x(x)$ of the data can then be obtained as

$$\hat{p}_{x;\theta}(x) = p_z(f_\theta(x)) \left| \det \frac{\partial f_\theta(x)}{\partial x^T} \right|. \tag{3}$$

We here focus on volume-preserving NFs [81,85], where the building blocks of the networks are chosen such that the determinant of the Jacobian is constant. Further details on the network architecture can be found in Appendix B.

To find the estimator $\hat{p}_{x;\theta}$ which best represents the data, we train the network by minimizing its negative log-likelihood $\mathcal{L}_{\text{ll}}$ via

$$\mathcal{L}_{\text{ll}}(\theta) = -\frac{1}{N}\frac{1}{|X|}\sum_{x\in X} \log(\hat{p}_{x;\theta}(x)) \tag{4}$$

given the dataset X [80]. We normalize this loss function by the number of dimensions of the dataset, i.e., the number of channels N. In our application, this training reliably captures the data distribution of the electrophysiological data, setting the foundation for the extraction of meaningful and interpretable latent variables in the next step. Details of the training procedure, including optimizer settings, are described in Appendix C.

IV. EXTRACTING MEANINGFUL LATENT VARIABLES WITH RECONSTRUCTION ERROR LOSS

While log-likelihood maximization enables the NF to accurately model the data distribution, it does not ensure that the latent variables are interpretable or organized in a meaningful way. To address this, we use an additional reconstruction error loss that enforces a hierarchical structure among the latent variables, ordering them by their relevance to data reconstruction. This hierarchy enables us to identify a small subset of latent variables that effectively describe the underlying data manifold.

To implement this, we map a data point x to the latent space $z = f_\theta(x)$, set all but the first l dimensions of z to zero and reconstruct the data via the inverse map:

$$\tilde{x}_\theta^{:l}(x) = f_\theta^{-1}\left(\sum_{i=1}^{l} f_{\theta;i}(x)\hat{e}_i\right), \tag{5}$$

where $\hat{e}_i$ is the base vector of the ith latent dimension. If the data lie on a l-dimensional manifold, then this reconstruction will be exact, $\tilde{x}_\theta^{:l}(x) = x$, for all data points $x \in X$. In general, the reconstruction error for l dimensions on the whole dataset is given by

$$R_l(\theta) = \frac{1}{|X|}\sum_{x\in X} \left|x - \tilde{x}_\theta^{:l}(x)\right|^2. \tag{6}$$

As the dimensionality is unknown, we average this reconstruction error over the number of dimensions l up to a cutoff N_r to obtain the full reconstruction error,

$$\mathcal{L}_{\text{rec}}(\theta) = \gamma_{\text{rec}}\frac{1}{N_r}\sum_{l=1}^{N_r} R_l(\theta), \tag{7}$$

where the prefactor γ_{rec} is a hyperparameter. A similar reconstruction error has been proposed by Bekasov and Murray [83]; in contrast to their work, we here use equal weights $p_l = 1/N$ for the reconstruction errors $R_l(\theta)$ in all l dimensions.

The effect of this additional loss term can be seen in Fig. 3 for a simple task. The task is constructed with two

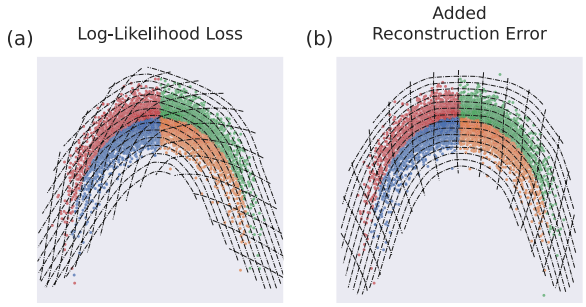


FIG. 3. Normalizing flow with reconstruction error learns inherent latent variables in a synthetic task. We illustrate the effect of adding the reconstruction error term using a synthetic dataset generated by sampling two independent Gaussian distributions: One for the radius $[r \sim \mathcal{N}(1, 0.1^2)]$ and one for the phase $[\phi \sim \mathcal{N}(0, (\frac{\pi}{3})^2)]$; data points are then given by $(r \sin\phi, r\cos\phi)$. Its topology naturally favors a description in terms of two latent variables corresponding to the two main axes, indicated here by different colors for the four quadrants of the data cloud separated by the two main axes. We train an invertible network $x = f_\theta^{-1}(z)$ on data points $x \in X$. In (a), the network is trained by maximizing the log-likelihood of x under the model [Eq. (4)], using Gaussian distributed latent variables $z \sim \mathcal{N}(0, \mathbb{I})$. In (b), an additional reconstruction error term is included in the loss function that encourages a small number of informative latent variables. For both cases, we show grid lines corresponding to the axes system $f_\theta^{-1}(s\,e_1 + t\,e_2)$ when varying only one of the two latent variables $s, t \in \mathbb{R}$ at a time. Steps between neighboring lines are half a standard deviation apart in the latent space. The lines are cut such that $\sqrt{t^2 + s^2} \leqslant 4$. By including the reconstruction error, the network correctly identifies the two inherent latent variables of the dataset: In (b), the grid lines align with the two main axes.

inherent latent variables: one corresponding to the radius and one corresponding to the phase. We visualize the data distribution in terms of drawn data samples and the learned distribution in terms of grid lines corresponding to the learned latent variables. Training a NF only on the negative log-likelihood [Eq. (4)] in Fig. 3(a) yields a good estimate of the data distribution. The latent variables of the learned distribution, however, do not match the inherent latent variables of the task. By including the reconstruction error [Eq. (7)] in the loss when training a NF [Fig. 3(b)], the network learns these inherent latent variables: radius and phase. For this task, we may even map the data to a lower-dimensional manifold: Setting the latent corresponding to the radius to zero, we obtain a one-dimensional curve that lies in the middle of the data points that still explains the data well. This two-dimensional example illustrates the principle to obtain a low-dimensional latent space.

More generally, the additional loss term [Eq. (7)] encourages a hierarchical structure of latent variables. This helps interpretability in the same way as sorting PC eigenvectors helps finding interpretable features of Gaussian statistics using PCA: by exploiting differences of scale. Different parameters tend to have different scales: In Fig. 3, r is drawn from a different distribution than ϕ and is, thus, mapped to a different direction in the latent space. If a distribution can be explained as n independent variables, that are transformed in some nonlinear way, then it is unlikely that two unrelated parameters have ranges of exactly the same size. Thus in a sense, it is the independence of the variables which confers their interpretability, since this allows the effect of each variable to be understood and manipulated separately.

Although simple, the two-dimensional example in Fig. 3 captures the essence of this general insight: It shows how we learn arbitrary distributions from data, align latent variables to interpretable axes, and thereby identify lower-dimensional manifolds by omitting the least informative dimensions.

V. MULTIPLE BEHAVIORAL STATES CAN BE CAPTURED BY MULTIMODAL LATENT SPACE

Most experimental datasets comprise data recorded within different behavioral states. This is typically reflected in a data distribution that has different modes or local maxima, as illustrated in Fig. 4(a). If the extent of each mode is not too large, then its center can be regarded as an archetypical data point that is representative of the corresponding behavioral state. Since the normalizing flows we are using here are volume preserving, the number of modes is the same in the latent space and in the data space [81,86].

To represent multimodal data distributions, we therefore use a Gaussian mixture distribution in the latent space [87], illustrated in Figs. 4(b) and 4(c). For the mixture components in latent space, we allow their centers to differ only in the first N_l dimensions—the center coordinates of the remaining $N - N_l$ dimensions are identical. Likewise, the variances of the modes may differ only in the same first N_l dimensions. This strategy is needed to prevent overfitting, i.e., setting the parameter N_l too high, the network does not generalize well to unseen data points. Both center coordinates and covariances are learned, while N_l is a hyperparameter that must be chosen *a priori*. Maximizing the log-likelihood alone results in the situation illustrated in Fig. 4(b): In the latent space, neither component alone suffices to distinguish the switch from one state to the other. Instead, the system of latent variables, represented by the vertical and horizontal axes, is rotated from the natural latent variables, indicated by the black lines. In contrast, including the reconstruction error as an additional loss term often leads to an alignment of a subset of latent variables with the natural latent variables of the system that drive the transition between behavioral states [Fig. 4(c)]. While this alignment is not guaranteed, it reflects a tendency for the model to organize latent variables in an interpretable manner, where changes between modes are associated with specific dimensions. Details on how the means and the covariances of the latent components are trained, together with their respective weights and on the decision to have $N_l = 10$ latent variables, in which the means and covariances differ, can be found in Appendix C.

VI. ESTIMATING THE NUMBER OF LATENT COMPONENTS OF THE DATA

Having introduced our normalizing flow framework, we now apply it to the neural activity data shown in Fig. 2. This requires choosing appropriate values for several hyperparameters, the most critical of which is the number of Gaussian mixture components in the latent space. This choice

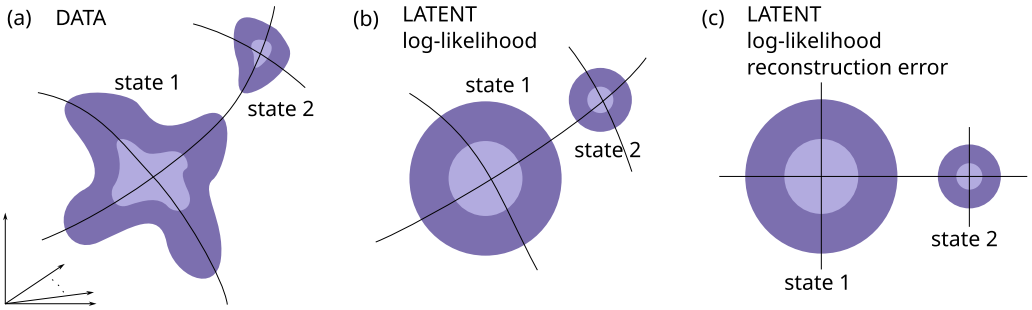


FIG. 4. Multimodal distributions in data and latent space. (a) A bimodal probability density in high-dimensional data space. The two modes may correspond to two distinct system states, such as different behavioral states. The coordinate system used by the data coordinates is shown in the bottom left, whereas the natural latent variables for these data are indicated as curved black lines. (b) Latent space with bimodal distribution when trained only with the log-likelihood loss term [Eq. (4)]. The dataset's natural latent variables do not align with the coordinate system chosen by this model. (c) Latent space with bimodal distribution when trained with both loss terms [Eqs. (4) and (7)]. The natural latent variables now align with the coordinate system [Eq. (7)].

determines the model's ability to represent distinct behavioral modes within the data.

Each component can be interpreted as a flat manifold in latent space, which is then mapped to a curved manifold in data space by the NF. The number of components significantly determines the achievable accuracy of the data description: For too few components, the description may be suboptimal and may miss relevant properties of the data, while with too many components, the manifolds start to overlap and thus become less interpretable.

To determine a suitable number of components, we train multiple instances of the network with different numbers of mixture components. The log-likelihood after training indicates the quality of the data description and is shown in Fig. 5 for up to 10 components. For comparison, we also show results from fitting linear GMMs. Their log-likelihood is overall worse than that of the NFs, indicating a less accurate representation of the data and thus a clear benefit of learning curved

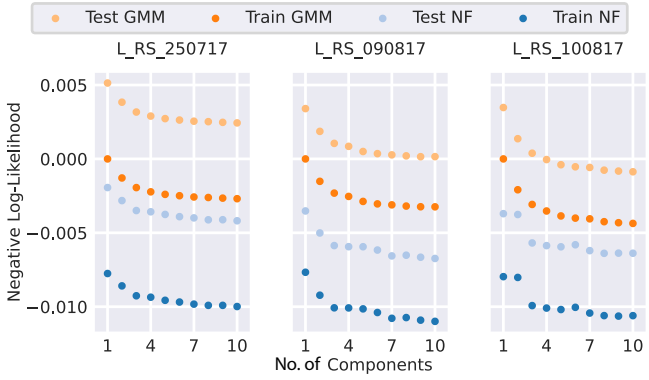


FIG. 5. Accuracy of data representation saturates with less than 10 latent components. For three different recording sessions, we show the negative log-likelihood per dimension for both train and test data is shown as a measure of representation quality. We compare NFs to linear GMMs to evaluate the advantage of learning a curved representation. NFs show a saturation at three to four latent components. GMMs perform overall worse for all numbers of components. An 80%-20% train-test split is fixed within the analyses of a recording session.

representations with NFs over linear methods like PCA. While the log-likelihood of the NFs monotonically increases with the number of components, it appears to saturate after three or four. Thus, NFs yield not only a more accurate representation of the data, but also require only a few mixture components. Also for the Neuropixels data from mouse cortex, we see an optimal performance already at a small number of components (Fig. 12).

To assess the learned data representations also qualitatively, we show in Fig. 6 the marginalized probability density in the first two PCA variables of the data for different numbers of mixture components in the latent space compared to the true marginalized distribution. To express the strong contrast between the two small, high-density regions and the vast, low-density region of the probability density, multiple components are necessary. From this visualization, we can also discern the reason for the smaller log-likelihood of GMMs compared to the NFs: The linear GMMs fail to account for the triangular cutoff of the probability density. Regions of high density, away from the triangular corners, are similarly well represented by the linear Gaussian mixture model and the representation continuously improves when increasing the given number of components without overfitting. The latter is expected, since a GMM with an arbitrary number of components is able to represent any empirical distribution. In contrast, the NFs are able to represent the triangular cutoff well even with few latent components due to their nonlinear mapping.

Based on these assessments, we consider normalizing flows with four latent components for the remainder of this paper. Furthermore, we focus on the first session (L_RS_250717) for the presentation of the results, as the results from the other sessions are comparable and consistent (see Appendix J).

VII. MIXTURE COMPONENTS OF NEURAL ACTIVITY CORRELATE WITH BEHAVIORAL STATES

In the previous section, we found that four latent components yield an accurate representation of the empirical data density for the electrophysiological data. If the multimodality of the data is partly due to the animal being in either of two

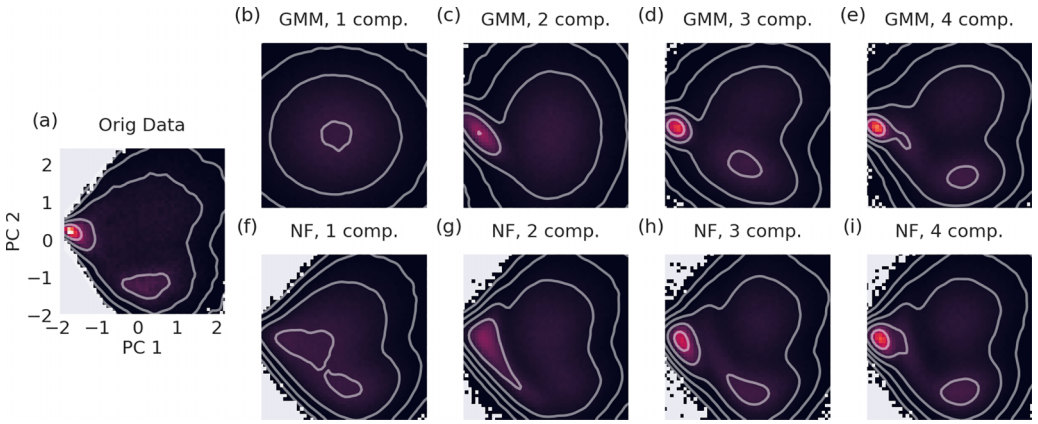


FIG. 6. Effect of number of latent components on the approximation capabilities: Marginalized distributions for the experimental data and learned models. (a) Marginalized distribution of one session of experimental data (L_RS_250717) along the first and second PCA directions. The color bar indicates the density of data points in the 2D histogram and applies to all panels. Light-gray isolines represent equal likelihood. [(b)–(e)] Marginalized probability densities for data sampled from GMMs with increasing numbers of latent components (one to four). [(f)–(i)] Corresponding distributions for NFs, also with one to four latent components.

different behavioral states—eyes open or eyes closed—then we would expect some mixture components to correlate with those states. And since the model has been trained without providing behavioral labels, this would indicate that states are distinguishable by structurally different neural activity.

To test this hypothesis, we compute the posterior probability of each component k given a neural data point x, denoted $p(k|x)$. This is obtained via Bayes's theorem:

$$p(k|x) = \frac{p_k p(x|k)}{p(x)} = \frac{p_k p(x|k)}{\sum_{k'} p_{k'} p(x|k')}. \tag{8}$$

Here p_k is the weight of the mixture component k, and $p(x \mid k)$ is the likelihood of the data point under that component. In a standard Gaussian mixture model, this likelihood would simply be $\mathcal{N}(\mu_k, \Sigma_k; x)$. However, since we use normalizing flows, we compute likelihoods using the change-of-variable formula:

$$\hat{p}_{x;\theta}(x|k) = \mathcal{N}(\mu_k, \Sigma_k; f_\theta(x)) \left| \det \frac{\partial f_\theta(x)}{\partial x^T} \right|, \tag{9}$$

where f_θ is the NF that maps data x into the latent space. These component-wise likelihoods, together with the priors p_k, yield the normalized posterior $p(k|x)$, which we interpret as the label probability for each component at a given time point.

We denote the time-varying label probabilities as $p(k|t) := p(k|x(t))$ and apply temporal smoothing with a Gaussian kernel ($\sigma = 3$ s) to obtain smoothed probabilities $q(k|t)$. Figure 7(a) shows these smoothed label probabilities as solid curves. To assign a dominant component at each time point, we compute $\mathrm{argmax}_k q(k|t)$, which is indicated by background shading in the figure.

Comparing the dominant components in Fig. 7(a) with the true behavioral states in Fig. 7(b), we observe that components C1 and C2 correlate strongly with the eyes-closed and eyes-open states, respectively. Components C3 and C4, which have lower likelihoods across time, appear to account for residual variability or transitions between dominant states.

Figure 7(c) further supports this relationship by directly comparing the distribution of label probabilities across behavioral conditions. Each component's distribution is split by behavioral state, revealing a clear preference: C2 shows

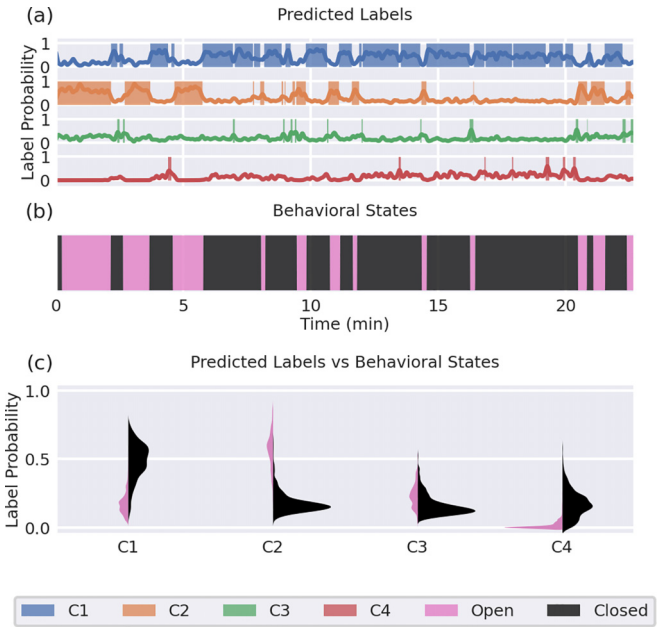


FIG. 7. Distinct mixture components in the latent space represent behavioral states. Components are ordered by mixture weight (C1 = highest weight). (a) For each of the four components (C1–C4), smoothed (Gaussian filter, $\sigma = 3$ s), normalized likelihoods are shown as solid curves. Labels are assigned by selecting the component with the largest label probability at each time point, indicated by background shading. (b) Ground-truth behavioral states—eyes open (pink) and eyes closed (black)—are shown for direct comparison with the labels predicted in (a). (c) The label probabilities are compared against the behavioral states. For each of the components, smoothed histograms display the distributions of label probabilities for the two behavioral states (left, eyes open; right, eyes closed).

high label probabilities predominantly during eyes-open periods, while C1 peaks during eyes-closed periods. Both these components are mostly indicative of the behavioral state. In contrast, C3 and C4 exhibit in general low probabilities, with C4 showing a clear preference for the eyes-closed state. These results confirm that the latent components learned by the model reflect meaningful behavioral distinctions in neural activity.

Since we observe this relation between mixture components and behavioral states consistently in all recording sessions (see Appendix J), we conjecture that our proposed method generally allows extracting latent components from neural data that correlate with behavioral states encoded in the neural activity. This capability extends beyond this particular dataset: The analysis of Neuropixels data also reveals that mixture components can capture stimulus-dependent states, with components showing preferential activation during visual stimulation, tactile stimulation, and baseline periods, respectively (Fig. 14).

VIII. ANALYTICALLY TRACTABLE NETWORK APPROXIMATION

The trained normalizing flow successfully captures complex neural population activity, but the mapping it implements is hardly interpretable due to the large number of trained parameters. To gain more insight, we therefore need to simplify the mapping to make it more amenable to interpretation. Although the analysis of previous sections relates high-level behavior to mixture components, here the analysis relates latent variables of a specific component to the geometric and statistical properties of the neural activity manifold.

To achieve this, we approximate the learned mapping for each component by a quadratic function. We found the quadratic to be a good balance between a reduction in expressivity and the ease of interpreting latent variables, all while retaining the latent space structure and the ability to utilize the learned log-likelihood by the approximation's link to its invertible origin.

For each latent component, we approximate the inverse mapping $f_\theta^{-1}(z)$ [Eq. (2)] by a quadratic mapping from the latent space z to the data space x as

$$x = q(z) = \sum_\alpha \left(\frac{1}{2} \sum_{i,j} A^\alpha_{ij} z_i z_j + \sum_i B^\alpha_i z_i + c^\alpha \right) \hat{e}^\alpha. \quad (10)$$

The upper indices α denote the dimensions of the data space with $\hat{e}^\alpha$ being its base vectors. Instead, the lower indices i, j refer to dimensions in the latent space. The parameters c^α, b^α_i, and A^α_{ij} represent the bias, the linear, and the quadratic terms, respectively. We omit here an additional index for each component for the sake of readability. Note that $A^\alpha_{ij} = A^\alpha_{ji}$ due to symmetry.

This approximation serves as the simplest nonlinear model, capturing quadratic interactions among the latent variables z_i. Remarkably, we find that the trained deep network can be well approximated with this quadratic function. A qualitative intuition for the validity of this approximation can be gained when comparing in Fig. 8 the upper column [Figs. 8(a)–8(d)] against the middle column [Figs. 8(e)–8(h)], where respectively the inverse map and its approximation are used. For a quantitative evaluation of the fitting procedure see Appendix F.

The observation that the quadratic function serves as a good approximation is nontrivial, because the space of functions that can be encoded by a deep network is far larger. Whether this approximation holds will in general depend on both the dataset and the network architecture. For the neural recordings studied here, it serves as a useful approximation, yielding valuable insights into the neural code as shown in the subsequent sections.

IX. BEYOND GAUSSIAN STATISTICS OF NEURAL COORDINATION

A better understanding of the neural code requires insight into both the statistical and geometrical structure of the neural data. These can be quantified in terms of cumulants and curvature, and we will be using the quadratic approximation to derive both of these measures from one joint model. Although the cumulants can be estimated easily from the data itself, this approach helps unite those two sides of the same coin: By deriving both geometric and statistical features from the same model, we ensure that the corresponding information within both measured quantities match. Moreover, the simple quadratic model may be helpful to assess the two different origins of non-Gaussian correlations: the departure from Gaussianity of the density on the latent space, as illustrated in Fig. 1(a), and the mapping from the flat latent space into the curved space spanned by the data, as illustrated in Fig. 1(c). In Appendix I, we sketch how our approach may be used to separate these contributions.

Starting with the derivation of the statistical features, we move beyond second-order (Gaussian) descriptions towards a full characterization of the statistical structure of the neural code. This is evident from the non-Gaussian statistics found in the data (Fig. 2). Indeed, the presence of higher-order interactions can play a crucial role in shaping population dynamics [88,89].

As an illustration, in Figs. 8(a)–8(d) we show the probability densities that the network learned for each of the four different components, marginalized onto the first two PCA variables. Note that component C2 aligns to the eyes-open state, while the other components mostly align to the eyes-closed state. Notably, component C4 represents the high-likelihood mode in the tip of the triangular structure, as observed in Fig. 2. Components C2 and C3 show a clear non-Gaussian structure.

The analytically tractable network approximation from the previous section enables us to compute not just means and covariances, but also third- and fourth-order cumulants for each latent component. These quantities are derived from the characteristic function of the approximate network mapping, providing a powerful lens into the statistical structure of the data encoded by the model. The statistics allows us to quantify the extent by which neural interactions deviate from Gaussianity within each component.

We obtain the moment-generating function [90,91] from the quadratic approximation [Eq. (10)] of the network

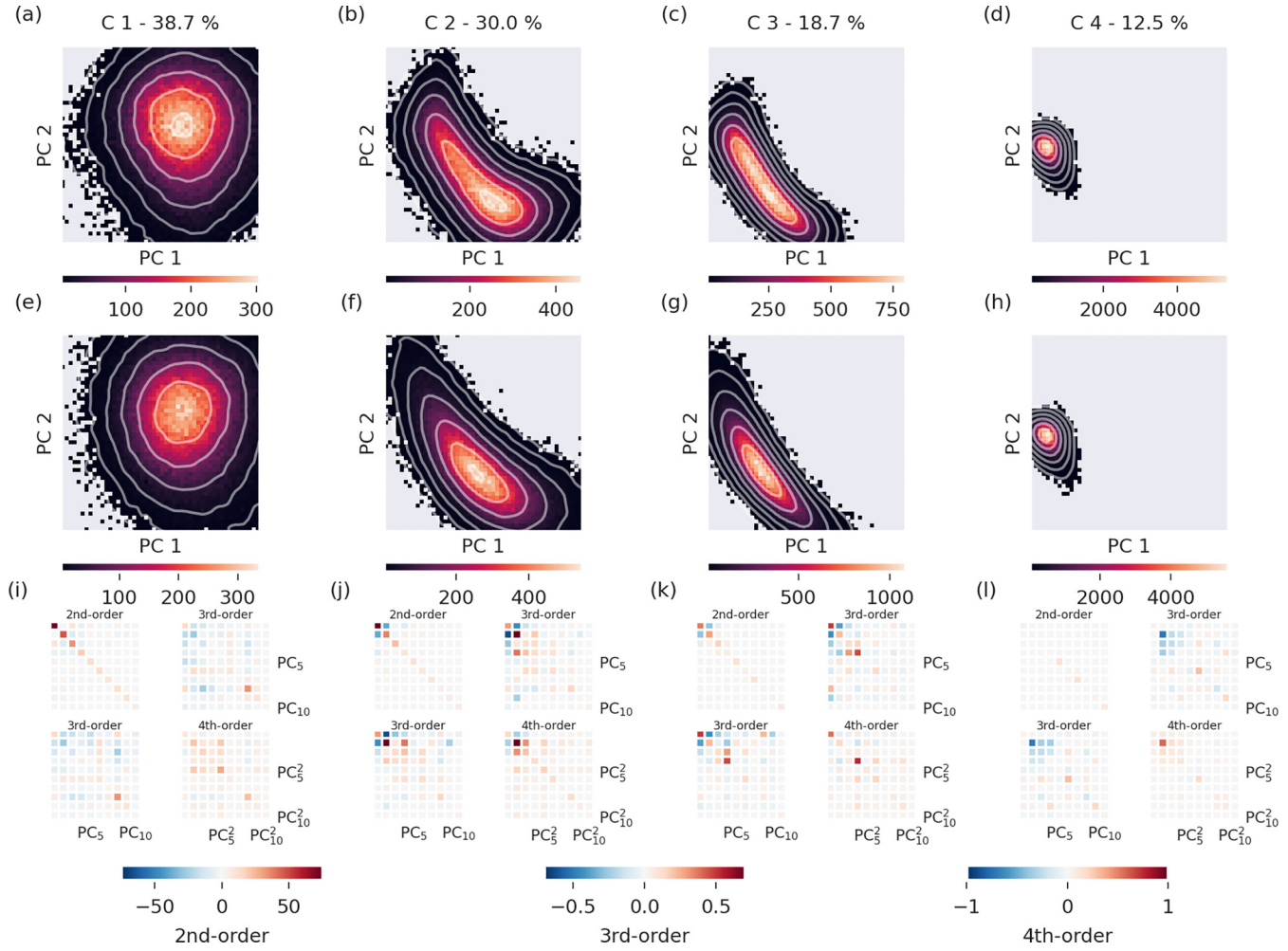


FIG. 8. Statistical characteristics of the latent components: Approximate marginalized distributions for experimental data and learned models. Components are ordered by mixture weight (C1 = highest weight). [(a)–(d)] Marginalized distributions of each latent component from the experimental session (L_RS_250717), projected onto the first two PCA directions. Light-gray contours indicate isolines of equal likelihood. The color bars indicate the density of data points in the 2D histograms. [(e)–(h)] Marginalized distribution of each component when the inverse mapping of the network is approximated by a quadratic function [Eq. (10)]. [(i)–(l)] Second- to fourth-order cumulants computed from this approximation.

mapping for each component as

$$Z(\boldsymbol{j}) = \frac{e^{j^T c} e^{\frac{1}{2} j^T B (\mathbb{I} - \sum_\alpha A^\alpha j^\alpha)^{-1} B^T j}}{\det\left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha\right)^{1/2}}. \tag{11}$$

Alternatively we may consider the cumulant-generating function, given by the logarithm of the moment-generating function. Derivations for both the moment- and cumulant-generating functions are given in Appendix G.

The resulting cumulants are an effective measure for non-Gaussianity of distributions, since for Gaussian distributions, all cumulants beyond second order vanish. We show the cumulants for all components in Figs. 8(e)–8(h); we follow here the same scheme as in Fig. 1 by showing all values of the second- order cumulant but only a reduced set of values for the third- and fourth-order cumulant.

As alluded to above, components C2 and C3 stand out by their pronounced third- and fourth-order cumulants, marking them as clearly non-Gaussian. Components C1 to C4 also show moderate higher-order structure.

These results demonstrate that the components inferred by our model exhibit distinct statistical signatures, with some showing strong non-Gaussian features over several dimensions, highlighting the importance of capturing higher-order structure to fully describe the neural population activity across behavioral states. The prevalence of non-Gaussian statistics extends also to the Neuropixels data from mouse cortex. They similarly exhibit pronounced third- and fourth-order cumulants across all latent components (Fig. 15), despite the fundamental differences in data characteristics (single-unit spikes vs multiunit envelopes, active stimulation vs resting state).

X. GEOMETRICAL PROPERTIES OF THE CURVED NEURAL MANIFOLD

To uncover the principles of neural coding, it is not only important to understand the statistical structure of neural activity but also to characterize the geometric shape of the population activity manifold. Neural activity patterns are

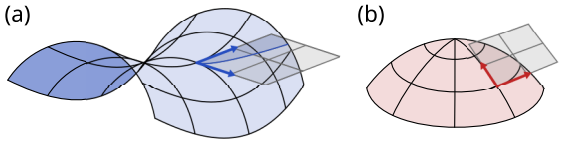


FIG. 9. Geometrical interpretation of data manifold curvature. Tangent vectors (arrows) define the local tangent space (gray) at a specific point on the data manifold. (a) At a saddle point, the tangent space intersects the manifold, resulting in negative sectional curvature. (b) For a spherical manifold, the tangent space lies entirely outside the surface, indicating positive sectional curvature.

often constrained to complex, nonlinear subspaces of the high-dimensional embedding space, and curvature is a key indicator of such nonlinearity. Having described the different manifolds in our data from a statistical perspective, we now change our perspective to a geometrical one. The geometrical perspective offers complementary insight into the structure of how the brain might organize neural states. It is also particularly relevant in light of the higher-order statistical dependencies identified in the previous section, as curvature can reflect underlying nonlinear interactions that are missed by second-order statistics.

Our method establishes a direct link between each learned latent component and a corresponding submanifold in the high-dimensional neural activity space. For the analyses presented here, we parametrize each component with $N_l = 10$ latent variables. We characterize the geometry of each mixture component separately by describing its curvature around its point of maximum likelihood.

To do this, we build on the quadratic approximation [Eq. (10)] of the network mapping after training, from which we construct local coordinate charts (see Appendix H). These charts form a smoothly varying atlas that enables the computation of intrinsic geometric quantities—the scalar curvature, which measures the overall average curvature, and the sectional curvature, which measures how the manifold bends in a particular point relative to a two-dimensional tangent plane. These concepts are central in differential geometry and follow standard definitions found in, e.g., Lee [92,93], Bär [94], and Robbin and Salamon [95]. These two measures reveal how the neural representations deviate from simple, flat geometries and provide insight into the complexity of the underlying neural code.

For the sectional curvature, we consider pairs of latent variables i and j and determine the curvature of the two-dimensional surface they span on the data manifold. To do so, we first define a local coordinate system on the data manifold in terms of tangent vectors e_i^α. These tangent vectors describe how perturbations in the latent variable z_i affect the data space x^α and are obtained from the quadratic approximation of the network model as

$$e_i^\alpha = B_i^\alpha + \sum_j A_{ij}^\alpha z_j. \tag{12}$$

Here B_i^α and A_{ij}^α are the linear and quadratic coefficients, linking latent variables to data dimensions. These vectors span the tangent space at a given point on the data manifold (see Chap. 3 in Ref. [92]), as illustrated in Fig. 9.

TABLE I. Scalar curvature for all four components.

	Sectional curvature
Component 1	−0.017
Component 2	−0.044
Component 3	−0.040
Component 4	−0.038

From these tangent vectors, we construct the metric tensor g_{ij}, which defines the inner product on the tangent space and encodes local geometric information such as distances and angles between directions (see chap. 2 in Ref. [93]):

$$g_{ij} = \sum_\alpha e_i^\alpha e_j^\alpha. \tag{13}$$

This metric tensor is essential in all further steps of the geometric analysis and enters also in terms of its inverse g^{ij}.

Next, we introduce the projection operator $P^{\alpha\beta}$ onto the normal space—the space orthogonal to the tangent plane:

$$P^{\alpha\beta} = \delta^{\alpha\beta} - \sum_{n,p} e_n^\alpha g^{np} e_p^\beta, \tag{14}$$

where $\delta^{\alpha\beta}$ is the identity matrix in data space. This operator ensures that the curvature is calculated with respect to deformations only perpendicular to the manifold.

Based on these definitions, we compute the sectional curvature $K(e_i, e_j)$ (see chap. 7 in Ref. [93]), which quantifies how the manifold bends in the plane spanned by the tangent vectors e_i and e_j as

$$K(e_i, e_j) = \frac{\sum_{\alpha,\beta} P^{\alpha\beta}\left(A_{ii}^\alpha A_{jj}^\beta - A_{ij}^\alpha A_{ij}^\beta\right)}{g_{ii}g_{jj} - (g_{ij})^2}. \tag{15}$$

We compute the scalar curvature R by averaging over all sectional curvatures at a point (see chap. 7 in Ref. [93]). For our model, this yields the expression:

$$R = \sum_{j,l,c,m} \sum_{\alpha,\beta} g^{jl} g^{cm} P^{\alpha\beta}\left(A_{jl}^\alpha A_{mc}^\beta - A_{cj}^\alpha A_{ml}^\beta\right). \tag{16}$$

The scalar curvature R provides a scalar invariant that summarizes the intrinsic curvature of the manifold at a point. In two dimensions, R coincides with the Gaussian curvature and reflects whether the sectional curvatures in the two PCA variable directions have equal or opposite signs. Positive curvature corresponds to elliptic (spherical-like) geometry, and negative curvature to hyperbolic (saddle-point-like) geometry (see Fig. 9). In higher dimensions, R generalizes this concept by averaging the sectional curvatures over all two-dimensional subspaces of the tangent space. A negative value of R indicates that, on average, the manifold bends away in multiple directions, although not all sectional curvatures need to be negative.

The detailed derivation of the expression for the scalar curvature can be found in Appendix H. We measure the scalar curvature in Table I for the different latent components: Component C2 exhibits the strongest negative curvature (–0.044), followed by component C3 (–0.040) and component C4 (–0.038), indicative of saddle-point-like manifold structures. In contrast, component C1 show a nearly flat geometry with a weak negative curvature (–0.017), suggesting

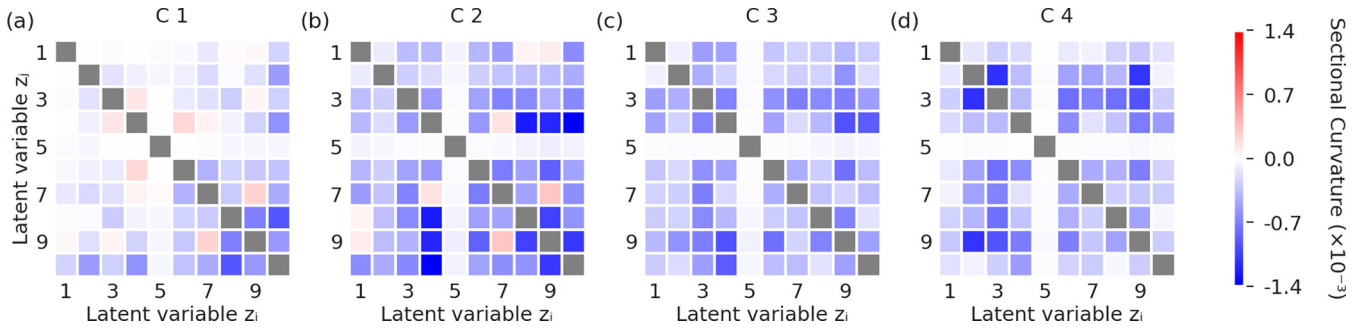


FIG. 10. Geometrical characteristics of the latent components. Components are ordered by mixture weight (C1 = highest weight). Panels (a)–(d) show the sectional curvature for each of the four components of the latent distribution. These values are computed from the quadratically approximated inverse network. Red indicates positive curvature, while blue indicates negative curvature. The color scale is consistent across all panels, with a range defined by the values reported in Table I for the scalar curvature (i.e., the mean of sectional curvatures).

only minor deviation from flatness. As we saw in Fig. 7, component C1 is mostly aligned to the eyes-closed state. It is also a component with moderate higher-order correlations (Fig. 8). Here we now observe that it is also a component lying on a near-flat manifold. In contrast, component C2, aligning mostly to the eyes-open state (Fig. 7) corresponds to the most curved submanifold. This component also shows the strongest deviations from Gaussianity (Fig. 8). The two components (C3 and C4) which do not align clearly to one of the states but rather account for transitions between or periods in-between one of the states (Fig. 7) both show a saddle-point-like geometry.

A more detailed picture emerges from the sectional curvature matrices shown in Fig. 10. Here the curvature contributions from different pairs of latent variables are visualized. For components C2, C3, and C4, the sectional curvature is predominantly negative (blue-shaded entries), supporting the interpretation of a globally saddle-point-like structure. In component C1, however, the curvature is more balanced, with both weakly positive and negative values, aligning with its near-zero scalar curvature and suggesting a less distinctly curved manifold.

These results suggest that curvature is not confined to a few exceptional latent variables but is distributed across the entire latent space. In other words, the manifold's curvedness reflects joint interactions among multiple latent variables rather than isolated effects of a few variables.

In particular, we observe that the component exhibiting the strongest higher-order statistical correlations lies on the most curved manifold. This alignment between statistical complexity and geometric structure supports our claim that understanding neural coordination requires tools capable of capturing both non-Gaussian statistics and nonlinear geometry, whereas our method extracts both aspects from a joint framework. Remarkably, analysis of Neuropixels data reveals a qualitatively different geometric signature: All latent components exhibit large positive scalar curvatures, indicating elliptic rather than saddle-point-like manifold geometry (Table II). This stark contrast suggests that manifold geometry may reflect important differences in neural circuit organization, recording modality, or behavioral context.

XI. DISCUSSION

This study presents an integrated framework designed to analyze complex neural population activity, focusing on three key objectives: (1) to extract coordinated activity across neurons, described both statistically via correlations and geometrically as manifolds; (2) to identify a compact set of latent variables that capture this structure and potentially relate to behavior; and (3) to provide interpretable and analytical characterizations of both the statistical properties (using the characteristic function) and the geometric features (providing a set of charts and measuring the curvature of the manifold).

To achieve these objectives, we propose a method utilizing NFs that describes how coordinated neural activity represents information. The method provides insights from both a statistical and a geometrical perspective, both derived from an invertible neural network trained in an unsupervised manner on neural data. An analytical description of the learned network mapping then enables us to compute both the statistical cumulants—of arbitrary order—and the geometric curvature of the original data.

Our approach merges the expressive capabilities of neural networks with the clarity provided by straightforward analytic expressions. As a result, it enables the extraction of non-Gaussian statistical structure, nonlinear dimensionality reduction, and the identification of latent variables aligned with behavior. Applied to visual cortical areas V1 and V4 in macaques, our method reveals interpretable latent components that align with behavioral states, each associated with distinct non-Gaussian statistics and curved submanifolds in neural state space.

The invertibility of normalizing flows implies that they provide explicit expressions for the likelihood of the data under the model: They can therefore be trained using the true likelihood. This contrasts with other architectures, such as variational autoencoders [58], where the likelihood is only implicit and therefore cannot be analyzed directly. The use of an additional loss function enables us to distinguish between latent variables that span the neural manifold and others that are classified as noise dimensions. In addition, by choosing a multimodal latent distribution, we may distinguish between the different behavioral states in the neural code. Although the training is unsupervised, the resulting representations

still align with the original behavioral states observed in the experiment—indicating that our method can identify behavioral states solely from observed recordings.

We examine the statistical and geometric properties of the neural recordings of V1 and V4 in macaque recorded by Chen *et al.* [84]. Statistically, the presence of third- and fourth-order cumulants reveals strong non-Gaussian structure in the data, reflecting the nonlinear nature of interaction in the neural code. The significance of such higher-order structure has been recognized since early information-theoretic analyses of neural populations [96], and higher-order correlations significantly influence key population-level statistics like entropy, sparsity, and statistical heat capacity [97] and are important for complex neural computation [98]. Geometrically, the curvature analysis uncovers predominantly saddle-point-like manifolds, characterized by consistently negative scalar and sectional curvature. This reflects a globally organized, rather than fragmented, structure of the neural code. Our framework enables a rigorous comparison of these submanifolds across latent variables, offering a detailed understanding of how they contribute to the overall shape and coordination of neural activity.

Our curvature-based analysis connects to a growing body of work exploring manifold geometry in both theoretical and applied contexts. On the theoretical side, recent developments in diffusion geometry provide tools to quantify curvature, tangent structures, and dimensionality directly from data [99], complementing foundational treatments in Riemannian geometry and its application to deep generative models [100]. Curvature has also emerged as a powerful analytic tool in neuroscience. For example, Acosta *et al.* [101] estimate extrinsic curvature to characterize the shape of neural manifolds, focusing primarily on synthetic data and including only a small-scale application to experimental place cell recordings. Their method, however, focuses on a pure geometric analysis and does not try to describe higher-order statistical structure such as cumulants. In contrast, our approach learns a full probability density over population activity, enabling the extraction of higher-order correlations and a richer description of neural variability. Furthermore, we apply our method to large-scale recordings involving approximately 800 simultaneously recorded neurons, providing a population-level analysis of manifold geometry. Curvature plays a critical role in statistical analysis on manifolds, such as determining the uniqueness of geometric medians [102] or improving dimension estimation [103]. It has also inspired a different line of research when analyzing network configurations, where it has revealed differences in structural [104] and functional connectivity [105]. While these diverse efforts underscore the value of curvature as a geometric descriptor, our approach distinctively integrates curvature-based geometric insights with a statistical model of correlations while providing a simple, interpretable structure.

A number of earlier theoretical contributions have emphasized the importance of correlated variability in shaping population codes. The review by Azeredo da Silveira and Rieke [106] framed this in terms of the geometry of information coding. Related theoretical work has further highlighted how specific forms of correlation can impact information encoding [107,108] and how population activity may be better understood within a probabilistic, inference-based framework [109]. Our framework provides a concrete implementation of these ideas by learning both the statistical correlations and the geometry of neural manifolds from data.

Building on this foundation, recent studies have highlighted the inherently nonlinear nature of these manifolds, especially during behaviorally rich tasks. For instance, multiple studies [24–26,28,30,31] have shown across multiple species and brain regions that low-dimensional neural manifolds are not merely a useful abstraction but that their nonlinear curvature and structure carry functional relevance. (For a review refer to Ref. [40].) These findings support the relevance of our approach, where we also go beyond linear assumptions and explicitly extract the curved geometry of neural activity. A related perspective is offered by Chung and Abbott [110], who advocate for population-level geometric analyses as a powerful tool to study both biological and artificial networks. Most directly, our work complements recent findings by Chou *et al.* [111], who demonstrated that the curvature of neural manifolds is tightly linked to the efficiency of representational untangling during task performance. Our work contributes to this growing effort by providing an interpretable and flexible framework for identifying such curved manifolds and linking their geometry to meaningful aspects of neural computation.

To enable both the statistical and geometric analysis presented above, we approximate the inverse of the learned network mapping by a quadratic function. Although this choice may not seem obvious, Appendix F shows empirically that the statistics of the approximation closely match those of the network, even for higher-order, off-diagonal components. This quadratic form allows us to compute statistical descriptors such as cumulants analytically, as well as to characterize geometric features such as curvature from the same functional representation. The resulting unified description enables rigorous comparisons across latent variables and behavioral states.

The increasing availability of large-scale neural recordings has created a pressing need for methods that can link brain-wide activity to computation and behavior [112]. Addressing this challenge requires approaches that are both theoretically grounded and computationally scalable. With the targeted extensions to normalizing flows introduced in this work, our method is designed to efficiently model high-dimensional neural population activity. We applied it to recordings with approximately 800 simultaneously recorded channels, and its architecture is readily scalable to even larger datasets, such as those obtained from Neuropixels probes with thousands of channels. This makes our framework a timely and adaptable tool for extracting interpretable structure from large-scale neural data.

We conduct a sample complexity analysis on our results (Appendix E), demonstrating that our method provides reliable scalar curvatures even with substantially reduced datasets (25% of full data), suggesting computational efficiency for large-scale applications while maintaining the statistical power necessary for robust curvature estimation.

Beyond multiunit activity, we validate our framework on single-unit spike train data from Neuropixels recordings (Appendix D). Importantly, its ability to extract meaningful structure persists across the transition from continuous-valued

signals to binary spike trains, suggesting that the underlying manifold organization reflects fundamental principles of neural population dynamics rather than artifacts of specific recording techniques. The fact that it works across such different data types strengthens our confidence in the approach and hints that our method could work as a general tool for studying neural computation.

Scaling to such high-dimensional recordings naturally raises concerns about computational cost and model complexity. To address this, we adopt an architectural strategy similar to that of Cramer *et al.* [113], where only a subset of the input dimensions are processed by the flow. Specifically, after a fixed linear transformation based on the input covariance, we pass only the top N_n PCA variables through the flow, while the remaining dimensions are directly passed to the latent space. This design keeps the network size manageable and ensures efficient training and evaluation. However, it comes with the trade-off that potentially informative structure in the discarded dimensions is not learned by the NF, which may limit the achievable log-likelihood.

An important constraint for applying NFs to other datasets is the inability of NFs to model perfectly noise-free dimensions, where the log-likelihood would diverge [66]. In practical applications such as ours, this is not a concern due to the inevitable presence of noise in electrophysiological recordings. However, this limitation should be carefully considered when applying our method to synthetic or near noise-free data.

In our implementation, we further constrain the architecture by selecting all flow blocks to be volume preserving [81]. While this is not a fundamental requirement of NFs, it is an explicit design choice. Volume preservation prevents the network from learning diverging mappings and contributes to more stable training dynamics, as noted in Ref. [114]. Although this restriction may limit model expressivity [85], we compensate this by using a more sophisticated latent space, which also allows the separation of different behavioral states.

Some of the drawbacks described are inherent to NFs and may not be present in alternative architectures. Among deep generative models, variational autoencoders (VAEs) [58] represent a natural alternative to normalizing flows for modeling high-dimensional neural activity in an unsupervised and probabilistic manner. While VAEs do not provide exact likelihoods and require both an encoder and a decoder network, they enable flexible nonlinear dimensionality reduction and generative modeling by optimizing a variational bound. Extensions such as β-VAE [115] and similar VAE adaptations promote disentangled and interpretable latent spaces through regularization schemes like global and input-dependent priors [116], penalizing correlations of latent variables [117–119], maximizing mutual information [120], or applying spatial masks [121]. More recent work has explored structured latent spaces [65], mixtures of local VAEs for manifold learning [67], and geometry-aware training procedures [63,64].

There appears to be no fundamental obstacle to extending our conceptual approach to such architectures. The key methodological steps—including approximating the decoder with a quadratic function and using it to compute characteristic functions, cumulants, and curvature tensors—could in principle also be applied to VAEs or similar architectures. As long as the decoder admits a suitable analytic approximation, the statistical and geometric descriptors introduced in this work remain valid. This highlights that our core contributions are not specific to NFs but rather propose a general strategy for latent-variable models.

Looking ahead, we identify several promising directions for extending this work: applying the framework to larger and more diverse datasets, incorporating time-varying manifold structures, analyzing multimodal sensory representations, and refining the structure of the latent space.

Although we here apply the framework to electrophysiological recordings in the macaque visual cortex, its design is fully general. It can be readily applied to neural population data from other brain regions or species, offering a principled way to extract and interpret latent structure in high-dimensional neural recordings. A natural next step is to apply our approach to more diverse and complex electrophysiological datasets. This will allow us to probe whether similar structured manifolds emerge in neural systems, e.g., involved in decision-making, motor control, or memory. Likewise, datasets involving more naturalistic or complex behaviors—such as goal-directed navigation or learning—could reveal how neural manifolds change for more complicated tasks. Cortical activity tends to show statistics close to a Poisson process [122–124]. Such processes have non-Gaussian cumulants, because the cumulant-generating function of a Poisson variable x resulting from binning a Poisson process of rate λ and bin width Δt is $W(j) = \ln\langle e^{jx}\rangle = \Delta\lambda e^{j} - 1 = \Delta\lambda \sum_{n=1}^{\infty} \frac{j^n}{n!}$, which shows that all cumulants are identical to the mean $\Delta\lambda$. A more concise description of the statistics is in terms of normal-ordered cumulants [125], which capture departures from uncorrelated Poisson statistics. Our estimated higher-order cumulants may thus reflect either correlations genuinely beyond Poissonian variability or the intrinsic non-Gaussian nature of Poisson processes themselves. A possible further improvement of the presented approach is to employ the likelihood function of a Poisson variable in training the network, i.e., $\ln p(x|\lambda(z)) = x \ln\lambda - \lambda - \ln x!$, using the normalizing flow to learn $\lambda(z)$. This could help distinguish between population-level correlations and single-neuron Poissonian variability, potentially revealing more refined insights into the statistical structure of the neural code.

The framework is also applicable across species, enabling cross-species comparisons of neural coding principles.

The modeling of time-varying manifolds, in which the statistical and geometric structure of each latent component is allowed to evolve over time, would be particularly relevant for very long recordings or experiments conducted over multiple days, where animals may gradually become more familiar with the task or experimental environment. In such cases, tracking slow drifts or systematic changes in the manifold structure or higher-order cumulants could reveal how internal models, attention, or behavioral strategies are reshaped over extended timescales.

In multimodal perception, neural responses to combined stimuli—such as visual and auditory inputs or different features within the same modality—may superimpose in a

nonlinear manner, rather than summing linearly. Our approach could help disentangle these complex interactions by revealing how diverse sensory components are jointly embedded and structured within the neural manifold.

Finally, further development of the latent space structure may enhance the model's flexibility. Our present model uses a Gaussian mixture, which provides a clear and interpretable decomposition of the data into distinct components. While being effective for many datasets, certain processes might be better represented using circular latent structures—for instance, periodic variables seen in orientation tuning or cyclic movements like gait—wherein their underlying phases exhibit inherent circularity. Alternatively, mixtures of more flexible distributions—rather than only Gaussians—could yield fewer components with the same expressive power. Incorporating such structures into the model's latent space would allow it to represent a broader class of neural dynamics while preserving interpretability.

To summarize, our approach provides a powerful, general tool for revealing latent variables, statistical dependencies, and geometric structure in high-dimensional neural data, setting the stage for future advances in neural manifold analysis across brain areas, behaviors, and species.

ACKNOWLEDGMENTS

This project is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Grant No. 368482240/GRK2416 and by the German Federal Ministry for Education and Research (BMBF Grant No. 01IS19077A to Jülich and BMBF Grant No. 01IS19077B to Aachen). S.N. acknowledges funding from Israel Science Foundation Grant No. 1442/21. C.M. acknowledges funding from Next Generation EU, in the context of the National Recovery and Resilience Plan, Investment PE1, Project FAIR "Future Artificial Intelligence Research" (CUP G53C22000440006). The authors gratefully acknowledge the computing time granted by the JARA Vergabegremium and provided on the JARA Partition part of the supercomputer JURECA at Forschungszentrum Jülich (computation Grant No. JINB33). We thank the authors of Chen *et al.* [84] for publicly providing the electrophysiological data from their macaque monkey experiments. We also thank Aitor Morales-Gregorio and Anno C. Kurth for suggesting this dataset, providing advice on preprocessing and engaging in fruitful discussions. Additionally, we thank Björn Kampa for fruitful discussions and infrastructural support for electrophysiological recordings in mice.

DATA AVAILABILITY

The data that support the findings of this article are openly available [126].

APPENDIX A: PREPROCESSING OF DATA

We used publicly available 1024-channel electrophysiological recordings from macaque V1 and V4 during resting state, recorded from Chen *et al.* [84]. The data were acquired using 16 Utah arrays (1.5-mm electrodes; Blackrock Microsystems), with 14 arrays in V1 and 2 in V4, targeting primarily layer 5. Preprocessing followed the procedure detailed in Morales-Gregorio *et al.* [32], including the extraction of the MUAe through high-pass filtering (500 Hz), rectification, low-pass filtering (200 Hz), and downsampling to 1 kHz, with notch filtering at 50, 100, and 150 Hz to remove line noise. Electrodes with signal-to-noise ratio below 2 or identified as contributing to high-frequency crosstalk (following Morales-Gregorio *et al.* [32]) were excluded.

For the current analysis, we further downsampled the MUAe to 100 Hz using FIR-filter-based decimation, trimmed edge samples to remove boundary effects, and z scored each channel independently. The resulting normalized signals across all arrays were concatenated into a single matrix, forming the input for subsequent analysis.

APPENDIX B: NORMALIZING FLOW ARCHITECTURE

The normalizing flow learns the distribution of single time points (for all channels together) of the MUAe recordings. The architecture, implemented in PyTorch, begins with a PCA-based preprocessing transformation inspired by Cramer *et al.* [113], which serves two technical purposes while maintaining the full bijective nature of the model.

Before applying a nonlinear transformation on the data with the NF, we perform a single PCA rotation and scaling of the data. This serves two roles: (i) It identifies directions whose variance is extremely small and (ii) it provides an orthogonal basis in which these low-variance directions are isolated. Importantly, these low-variance dimensions are not discarded: Although their statistical structure is too small to be learned with the NF, they are still passed to the model via a variance-normalizing transformation. They therefore remain part of the overall bijection and contribute fully to the likelihood. Separating the dimensions in this way addresses two issues simultaneously. First, it avoids severe conditioning problems in the Jacobian that arise when combining dimensions whose variances differ by orders of magnitude. Second, and equally important, it significantly reduces computational cost, since otherwise the NF would spend resources (parameters, flow layers, or training time) on dimensions that contain no meaningful structure. The saved capacity allows the flow to allocate more expressive power to the high-variance directions.

The main part of the nonlinear flow comprises 10 volume-preserving blocks, each consisting of a linear all-to-all mixing layer followed by an additive coupling layer [80], using ELU activations and a hidden dimension of 128. All components are invertible, enabling exact likelihood computation and efficient sampling.

APPENDIX C: TRAINING PROCEDURE

The training process consists of pretraining and main training. Prior to training, model parameters are initialized to promote numerical stability and effective convergence. The single PCA-like transformation projects the input onto the eigenbasis of its covariance matrix and normalizes it to unit variance, based on the training data. This layer stays constant over the training. The linear all-to-all and the additive coupling layers are initialized using orthogonal matrices for the

linear layers and biases are sampled uniformly from the interval $[-1/N_n, 1/N_n]$, where N_n is the number of dimensions that is processed by the main building blocks. This initialization ensures proper signal propagation and variance preservation through the network.

For the initial latent distribution, a GMM with four components is fit to a projection of the training data after the PCA-like transformation into a $N_l = 10$-dimensional latent space. We use the same value here as for $N_r = 10$ [see Eq. (7)]. These low values are chosen so that the analysis of cumulants and curvature can be displayed more clearly.

The GMM is implemented using scikit-learn's *GaussianMixture* model and is refit at the beginning of each main-training round based on the current latent output. Each component k of the GMM is defined by a weight p_k, a mean vector μ_k, and a covariance matrix Σ_k. These parameters are estimated by the GMM fitting procedure. By design, μ_k is nonzero only in the first N_l dimensions, and Σ_k is constrained to deviate from the identity only in the top-left $N_l \times N_l$ block. These constraints also hold for the refitting in the main training.

In the pretraining stage, the network is trained for 100 epochs using the standard likelihood loss function [Eq. (4)].

The main-training consists of 20 rounds of five epochs each. Here the reconstruction error loss [Eq. (7)] is added. The weight is chosen to be $\gamma_{\text{rec}} = 0.02$. At the beginning of each round, a new GMM is refit on the current latent output. The network parameters are continuously trained throughout the training rounds.

Optimization in both phases uses the Adam optimizer with a learning rate of 0.001, βs (0.9, 0.999), ε of 1e-8, and no weight decay. A batch size of 10 000 time points is used throughout training.

APPENDIX D: NEUROPIXELS DATA

We analyzed parallel *in vivo* electrophysiological recordings obtained with Neuropixels probes from the mouse visual cortex (V1) and superior colliculus (SC). Mice were awake, head-fixed and passively stimulated, performing no behavioral task. Sixteen-week-old male mice were on a rubber-coated running wheel allowing leg movement, and Neuropixels probes [127] were inserted into the left hemisphere of the brain. Surgical procedures for the electrophysiological recordings are described in Refs. [128]. The bottom 384 channels of the Neuropixels probe were used to record high-pass–filtered action potential signals at 30 kHz. Voltage traces from each channel were processed using Kilosort 2.5 [129] to detect and cluster spike waveforms corresponding to individual neurons.

Visual stimuli consisted of Gaussian noise movies (60 fps, 100 ms duration) presented contralaterally at a distance of 17 cm from the right eye. Tactile stimulation comprised 50 ms air puffs (0.03 bar) delivered contralaterally to the whiskers, inducing deflections without startle responses. Each stimulus type was presented twice per trial for 40 trials, with one trial lasting 3.6 s.

We analyze the activity of neurons from two brain regions: $N = 28$ neurons from primary visual cortex (V1) and $N = 65$ neurons from SC, totaling 93 simultaneously recorded units. The data were concatenated across trials to form a continuous time series (trials $\times$ time_bins $\times$ neurons), then reshaped into a 2D matrix (time_points $\times$ neurons) for input to the normalizing flow.

Stimulus labels were encoded as a three-state system: visual stimuli, tactile stimuli, and no stimulus periods. We determined the stimulus presentation window to be 0.9–1.9 s in each trial.

To smooth the inherent noise in binary spike trains while preserving temporal dynamics, we applied Gaussian filtering with a 100-ms window along the temporal dimension. This temporal smoothing helps reveal underlying population dynamics while maintaining the discrete nature of spike train data. Following filtering, we added reproducible uniform noise (1% of the filtered signal's standard deviation) to prevent numerical issues during training.

All experiments complied with the German Animal Protection Law and were approved by the local ethics committee (LANUV, NRW). Spike trains were binned at 5 ms resolution, with all nonzero entries set to 1. Trials were concatenated into a single matrix for subsequent analysis.

1. Configuration differences for Neuropixels analysis

The Neuropixels analysis requires some adjustments compared to the main dataset. Most notably, we reduced the dimensionality of the nonlinearly processed dimensions from $N_n = 70$ to $N_n = 20$ to accommodate the smaller neural population (93 vs 774 channels). The temporal resolution was increased to 5 ms binning compared to the 10 ms binning used for macaque data. We also reduced the reconstruction loss weight from $\gamma_{\text{rec}} = 0.02$ to $\gamma_{\text{rec}} = 0.01$ These parameter adjustments were necessary to optimize for the distinct characteristics of the spike train data.

2. Higher-order correlations in Neuropixels recordings

In Fig. 11, we show spike raster plots with structured activity patterns across the 93 simultaneously recorded neurons, with clear responses to both visual and tactile stimuli. Principal component analysis reveals that the first few components capture most of the variance, indicating low-dimensional structure in the population activity.

Notably, the marginal distributions of individual principal components show pronounced deviations from Gaussianity [Fig. 11(c)], and the 2D marginalization exhibits a complex, elongated structure that differs markedly from the triangular shape observed in the macaque data. The presence of significant third- and fourth-order cumulants [Figs. 11(e) and 11(f)] confirms structured non-Gaussian correlations across the neural population.

3. Model performance and component selection

Normalizing flows achieve substantially better likelihood scores than linear Gaussian mixture models across all component numbers (Fig. 12). This superior performance demonstrates the importance of capturing nonlinear manifold structure in spike train data. The marginalized distributions (Fig. 13) show that NFs capture the complex, elongated structure of the Neuropixels data more accurately than GMMs, particularly the irregular boundaries and multimodal features.

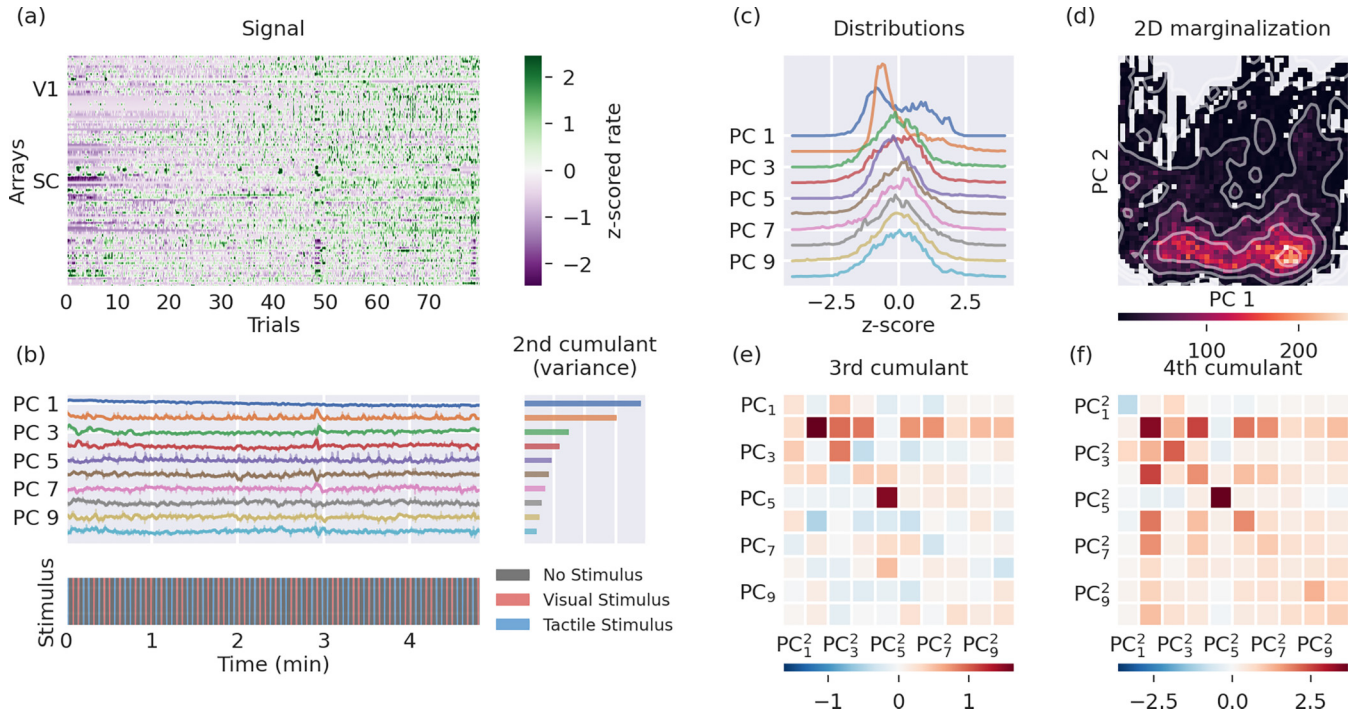


FIG. 11. Neuropixels data exhibit higher-order correlations and stimulus-dependent neural activity. (a) Spike raster from simultaneous recordings of 93 neurons (28 V1, 65 SC) during visual and tactile stimulation paradigm. Each trial lasted 3.6 s with stimuli presented at 1 s and 1.5 s. Spike trains were binned at 5-ms resolution with binary encoding. (b) Time evolution of the first 10 PCs of the z-scored data. The solid curves show temporal smoothing of the raw PCs using a Gaussian kernel. The stimulus barcode below indicates visual stimuli (red), tactile stimuli (blue), and no stimulus periods (gray). The rapid decay of the eigenvalues (variance; right panel) indicates that the neural data lie on a lower-dimensional manifold. (c) Marginal distributions of the first 10 PCs. While some appear Gaussian, others clearly deviate from Gaussianity. (d) Two-dimensional marginalization along the first and second PCs. [(e) and (f)] Third- and 4thfourth-order cumulants. For both cases, the signals along the PCs are first divided by their standard deviation σ_i to normalize for amplitude. Thus, panel (e) portrays x_i/σ_i against x_j^2/σ_j^2 and panel (f) x_i^2/σ_i^2 against x_j^2/σ_j^2. Color bars indicate cumulant magnitudes. The presence of nonzero higher-order cumulants demonstrates structured non-Gaussian statistics in the Neuropixels data.

In contrast to the Gaussian mixture model, the likelihood doesn't notably increase using further components on this dataset. Although we have three known labels, we nevertheless selected the four component model for further analysis: In addition to more closely matching the analysis of the previous dataset, that previous analysis also revealed that having more components than labels helps capture both the label structure and potential remaining noise.

4. Stimulus-dependent neural states

The learned mixture components show clear alignment with experimental conditions (Fig. 14). Component C2 exhibits preferential activation during tactile stimulation periods, while Component C3 shows stronger responses to visual stimuli. Components C1 and C4 appear to capture baseline activity and transitions between stimulated states. This unsupervised discovery of stimulus-dependent neural states demonstrates the method's ability to extract meaningful structure also for the spike train data.

5. Manifold geometry and curvature

The geometric analysis reveals manifold structures that differ qualitatively from those observed in macaque visual cortex. All four components exhibit large positive scalar curvatures (Table II): C1 (R = 5375), C2 (R = 873), C3 (R = 665), and C4 (R = 1429). These positive values indicate elliptic (spherelike) rather than hyperbolic (saddle-point-like) manifold geometry. It is worth mentioning that it is not as straightforward to interpret the magnitudes of the scalar curvatures, which differs in order of magnitude from the macaque visual cortex data.

TABLE II. Scalar curvature values for all four latent components in Neuropixels data. All components exhibit large positive scalar curvatures, indicating elliptic (spherelike) manifold geometry. Component C1 shows the highest curvature, followed by C4, C2, and C3. These positive values contrast sharply with the negative scalar curvatures observed in macaque visual cortex data, suggesting fundamentally different manifold structures between recording modalities.

	Sectional curvature
Component 1	5375
Component 2	873
Component 3	665
Component 4	1429

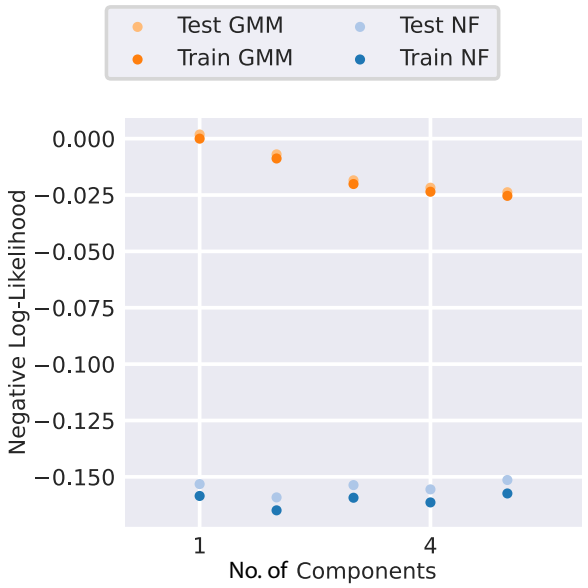


FIG. 12. Accuracy of data representation saturates with less than five latent components for Neuropixels data. The negative log-likelihood per dimension for both train and test data is shown as a measure of representation quality. We compare NFs to linear GMMs to evaluate the advantage of learning a curved representation. NFs achieve substantially better likelihood scores than GMMs across all component numbers. The superior performance of NFs demonstrates the importance of capturing nonlinear manifold structure in Neuropixels spike data. An 80%-20% train-test split is used for model evaluation.

The sectional curvature analysis (Fig. 16) shows mixed positive and negative values across latent variable pairs, but the overall positive scalar curvatures suggest different manifold structures compared to the macaque data, where negative scalar curvatures were consistently observed. This contrast may be due to a number of reasons: the recording modality (single-unit spikes vs multiunit activity), brain region (V1 and SC vs V1 and V4), the animal type (mouse vs macaque), and experimental paradigm (active stimulation vs resting state) are different.

6. Implications for neural manifold analysis

The application to Neuropixels data demonstrates the broad applicability of our approach for different recording modalities and species. The key findings include: The method is able to capture non-Gaussian correlations and curved manifold structure in binary spike train data. Unsupervised component discovery aligns with experimental stimulus conditions. And the manifold geometry differs clearly between recording modalities.

These results support our claim that understanding neural population activity requires tools capable of capturing both non-Gaussian statistics and curved geometry.

APPENDIX E: SAMPLE COMPLEXITY ANALYSIS

An assessment of the geometric analysis method requires quantifying the sensitivity of the estimated quantities to the amount of data used for model training. This is important to show the robustness and stability of the two-step fitting procedure (Sec. VIII), which relies on a quadratic approximation of the inverse normalizing flow.

To characterize the sample complexity for our model, we systematically vary the fraction of the full dataset used for the pipeline, in which the model is trained and the subsequent geometric analysis is performed. We test sample fractions of 25%, 50%, 75%, and 100% of the available samples from session L_RS_250717. To quantify the variability associated with the training process, the entire analysis pipeline is executed for 10 independent runs at each specified sample fraction.

1. Stability and convergence

The results of these experiments, presented in Fig. 17, demonstrate the stability and convergence of the estimates of the scalar curvature across different sample sizes. For scalar curvatures [Fig. 17(a)], the mean values averaged across the 10 runs tend to converge to stable values as the sample fraction increases.

Reducing the dataset size available to the training of the NF leads to less pronounced scalar curvatures in the learned components, although their main characteristics, in particular, relative sizes and the sign of the curvature, remain stable.

Component weights [Fig. 17(b)] demonstrate stability and minimal variability across the entire sample size range. The small magnitude of the error bars across all sample fractions underscores the robust estimation of the component weights and indicates that the mixture decomposition remains stable even with substantially reduced training data.

Figure 18 shows the test log-likelihood as a function of sample fraction, where each trained model is evaluated on its corresponding test split (20% of data are reserved for the test set at each run). The worse likelihood with reduced dataset size can be understood as a driving factor for the observed changes in the scalar curvature. The test log-likelihood loss gets systematically worse as the training sample fraction is reduced, from approximately 1.069 at 100% sample size to 1.082 at 25% sample size. This change in test performance represents a meaningful but modest degradation, suggesting that the normalizing flows maintain reasonable generalization capability even when trained on reduced datasets.

The larger error bars and increased scatter at smaller sample fractions (particularly visible at 25%) reflect increased variability in model performance when training data are limited, consistent with the principle that smaller datasets lead to less stable model fitting. However, the relatively small magnitude of performance degradation indicates that reliable density modeling can be achieved across the range of sample sizes tested.

2. Theoretical considerations

The precision of curvature estimates fundamentally depends on accurate estimation of the quadratic coefficients A_{ij}^{α} from finite samples. This estimation challenge aligns with established results from tensor estimation literature [130–132], where one aims to estimate a rank-p tensor composed of M

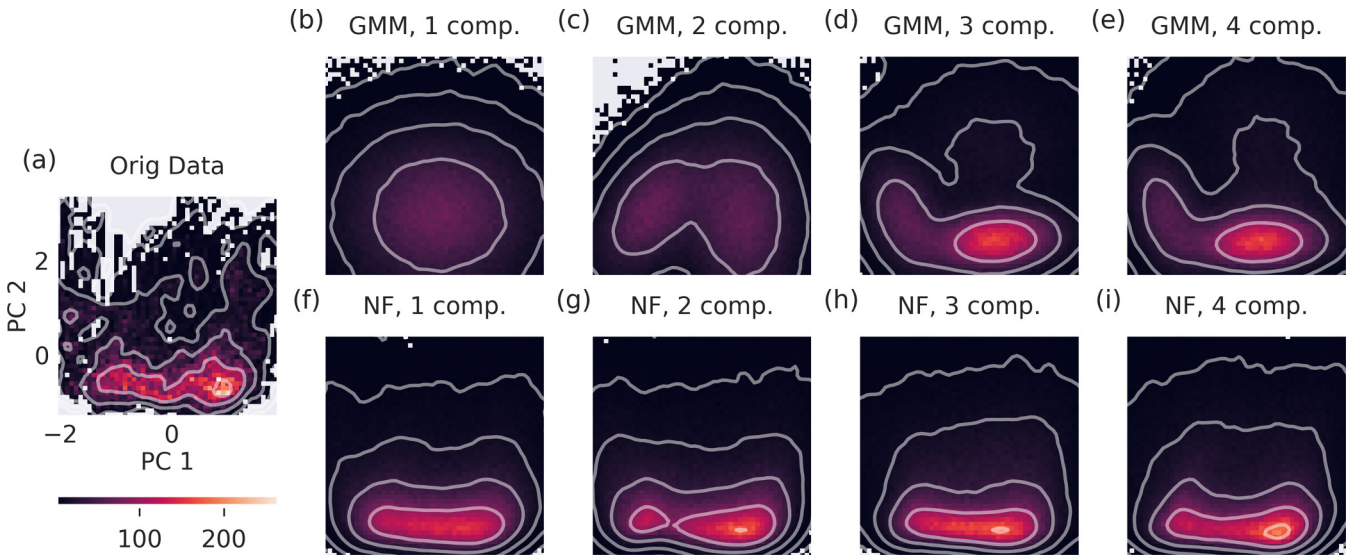


FIG. 13. Effect of number of latent components on approximation capabilities for Neuropixels data. Marginalized distributions along the first and second principal components for experimental data and learned models. (a) Marginalized distribution of Neuropixels data projected onto the first two PCA directions. Light-gray isolines represent equal likelihood contours. [(b)–(e)] Marginalized probability densities for data sampled from GMMs with increasing numbers of latent components (one to four). [(f)–(i)] Corresponding distributions for NFs, also with one to four latent components. NFs capture the complex, elongated structure of the Neuropixels data more accurately than GMMs, particularly the irregular boundaries and multimodal features visible in the original data. The superior performance of NFs shows their ability to model the nonlinear manifold structure inherent in spike train data.

orthonormal vectors e^μ (called spikes) in the following way:

$$T_{i_1 \dots i_p} = \sum_{\mu=1}^{M} c^\mu e^\mu_{i_1} \dots e^\mu_{i_p},$$

where c^μ are scalar prefactors. The estimation challenge is then to recover the vectors e^μ from a noisy estimate of $T_{\text{observed}} = T_{\text{true}} + \text{noise}$.

In our case, one can show that if A has the same low-rank structure, then one can recover it from the third cumulant of the data. Hence, if we can estimate a low-rank decomposition of $\langle\langle x^\alpha x^\beta x^\gamma \rangle\rangle$, then we can find A. Given a finite number of samples x, however, we only have access to a noisy estimate of the third cumulant:

$$\langle\langle x^\alpha x^\beta x^\gamma \rangle\rangle_{\text{observed}} = \langle\langle x^\alpha x^\beta x^\gamma \rangle\rangle_{\text{true}} + \text{noise}, \qquad \text{(E1)}$$

where the noise term scales as $1/\sqrt{P}$ with sample size P.

The aforementioned theoretical works prove bounds for sample complexity in the case of low-rank tensor recovery problems. Under idealized conditions with low-rank structure and Gaussian noise, the results suggest sample complexities scaling as $P \sim d^{p-1}$ for pth order tensors in d dimensions using Langevin dynamics [131], or $P \gg d^{p-2} \log d$ for stochastic gradient descent [132]. For third-order tensors ($p = 3$) relevant to curvature analysis, this implies quadratic or linear times logarithmic scaling with ambient dimension.

However, these theoretical bounds represent best-case scenarios under specific low-rank assumptions that may not fully apply to our setting. Our analysis neither tackles tensor recovery nor assumes low-rank structure in the underlying generative process. Nevertheless, these results provide a valuable perspective on the computational challenges of higher-order statistical estimation from finite samples, and thus give us an intuition on how well the terms A^α_{ij} can be learned with finite samples.

3. Conclusion

The dependencies of the scalar curvatures, the component weights, and the log-likelihoods on the sample size validate the reliability of the our method and the two-step fitting procedure. We can conclude that qualitative geometric insights can be reliably obtained with as little as 25% of the data, while quantitative precision needs larger sample sizes.

APPENDIX F: VALIDITY OF QUADRATIC APPROXIMATION

In this Appendix, we describe the procedure used to approximate the inverse mapping of each latent component by a quadratic function and validate it. This approximation is intended to simplify the interpretation of the complex mapping implemented by a trained normalizing flow.

The procedure starts from the latent components which are a result of the training procedure (see Appendix C). Each component k is characterized by its mean μ_k and covariance Σ_k. Samples z drawn from these components are mapped back to data space via the inverse flow, $x = f_\theta^{-1}(z)$.

For each mixture, we perform a quadratic fit to approximate the inverse mapping from latent space z to data space x. The steps involved are as follows:

1. Feature construction

For each latent mixture, features are constructed by including both linear terms (z_i) and quadratic terms ($z_i z_j$) to be able

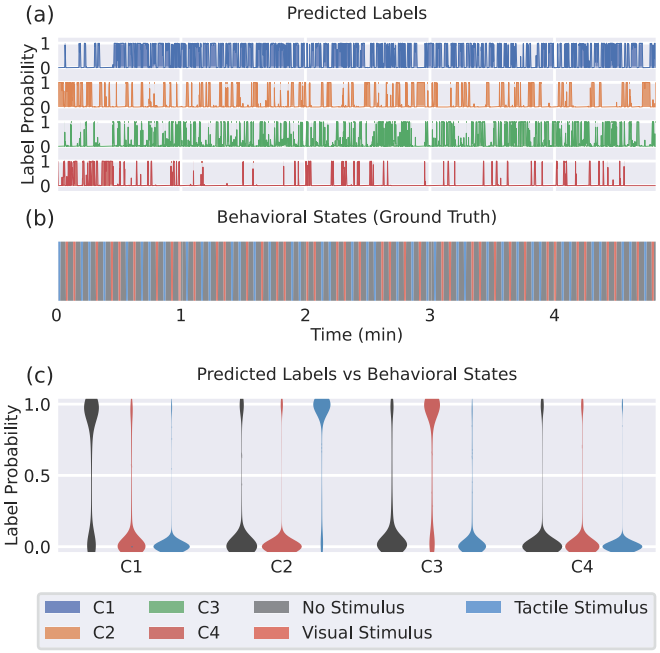


FIG. 14. Distinct mixture components in the latent space represent stimulus-dependent neural states in Neuropixels recordings. (a) For each of the four components (C1–C4), smoothed (Gaussian filter, $\sigma = 3$ s), normalized likelihoods are shown as solid curves. Labels are assigned by selecting the component with the largest label probability at each time point, indicated by background shading. (b) Ground-truth stimulus conditions—visual stimuli (red), tactile stimuli (blue), and no stimulus periods (gray)—are shown for direct comparison with the labels predicted in (a). (c) The label probabilities are compared against the stimulus conditions. For each of the components, smoothed histograms display the distributions of label probabilities for the three stimulus conditions (left, no stimulus; center, visual stimulus; right, tactile stimulus). Component C1 shows a clear preference for the no stimulus state. Component C2 shows preference for tactile stimulation periods, while Component C3 exhibits stronger activation during visual stimulation. Component C4 appear to capture baseline activity and transitions between stimulated states.

to fit all coefficients of Eq. (10). The feature construction is achieved using *PolynomialFeatures* from scikit-learn.

2. Model fitting

A *LinearRegression* model is used to fit these polynomial features against the corresponding data mixtures. This fitting captures both linear and quadratic interactions among latent variables. The parameters obtained from this fitting process—bias c^α, linear terms B_i^α, and quadratic terms A_{ij}^α—are crucial for understanding how changes in latent variables affect neural population activity.

Empirical analysis (see Figs. 19–21) reveals that the cumulants up to order 4 are well approximated by the quadratic function across all components: Although deviations appear, especially for higher-order off-diagonal components, the approximated statistics remain strongly correlated with the true ones. This observation is significant as it indicates that despite the reduced complexity of the quadratic model compared to the full network, it retains essential statistical properties of neural activity. The close alignment between original and approximated cumulants suggests that key interactions among latent variables are preserved, providing confidence in using this simplified model for further interpretation and analysis.

APPENDIX G: CHARACTERISTIC FUNCTION AND CUMULANTS

To compute the cumulants of the data variables x^α, we approximate the trained normalizing nlow f_θ^{-1} by a second-order (quadratic) function of the latent variables z (G1). For each mixture component, the latent variables follow a multivariate Gaussian distribution. This structure allows for the analytical evaluation of the moment generating function $Z(\boldsymbol{j})$ (characteristic function), and from it, the cumulant generating function $W(\boldsymbol{j}) = \ln Z(\boldsymbol{j})$. By expanding $W(\boldsymbol{j})$ in powers of the source vector $\boldsymbol{j}$, the cumulants can be obtained as derivatives evaluated at $\boldsymbol{j} = 0$. The derivation uses standard results from Gaussian integrals, matrix calculus, and series expansions to express each cumulant in terms of the parameters A^α, B^α, and c^α.

The starting point for the calculations of the cumulants is the approximation of the network as a quadratic mapping from latent space to data space [Eq. (10)],

$$x^\alpha = \frac{1}{2}\sum_{i,j} A_{ij}^\alpha z_i z_j + \sum_i B_i^\alpha z_i + c^\alpha. \tag{G1}$$

Without loss of generality, we can always transform the latent variables $z \sim \mathcal{N}(\mu, \Sigma)$ such that $\tilde{z} \sim \mathcal{N}(0, \mathbb{I})$. This standardization simplifies the analytical treatment and allows us to carry out all computations using standard Gaussian identities. For each mixture component k, this transformation effectively replaces the parameters as

$$\begin{aligned}
\tilde{c}_k^\alpha &= c^\alpha + B^{\alpha T}\mu_k + \tfrac{1}{2}\mu_k^T A^\alpha \mu_k \\
\tilde{B}_k^\alpha &= \Sigma_k^{1/2} B^\alpha + \Sigma_k^{1/2} A^\alpha \mu_k \\
\tilde{A}_k^\alpha &= \Sigma_k^{1/2} A^\alpha \Sigma_k^{1/2}.
\end{aligned}$$

With a slight abuse of notation, we omit the tilde in this renaming in the derivation for legibility, where the index k stands for the Gaussian component here.

We now use the expression above to compute the moment generating function, averaging over the standard Gaussian distribution $z \sim \mathcal{N}(0, \mathbb{I})$

$$\begin{aligned}
Z(\boldsymbol{j}) &= \left\langle e^{\sum_\alpha j^\alpha x^\alpha} \right\rangle_z \\
&= \left\langle e^{\sum_\alpha j^\alpha c^\alpha + \sum_{\alpha,i} j^\alpha B_i^\alpha z_i + \frac{1}{2}\sum_{\alpha,i,j} j^\alpha A_{ij}^\alpha z_i z_j} \right\rangle_z \\
&= e^{\sum_\alpha j^\alpha c^\alpha} \int d^n z \frac{1}{(2\pi)^{n/2}} \Big(e^{-\frac{1}{2}\sum_{i,j} z_i \delta_{ij} z_j} \\
&\quad \times e^{\sum_\alpha j^\alpha B_i^\alpha z_i + \frac{1}{2}\sum_{\alpha,i,j} j^\alpha A_{ij}^\alpha z_i z_j} \Big).
\end{aligned}$$

We define a $\boldsymbol{j}$-dependent covariance matrix Σ as

$$\Sigma(\boldsymbol{j}) := \left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha \right)^{-1}.$$

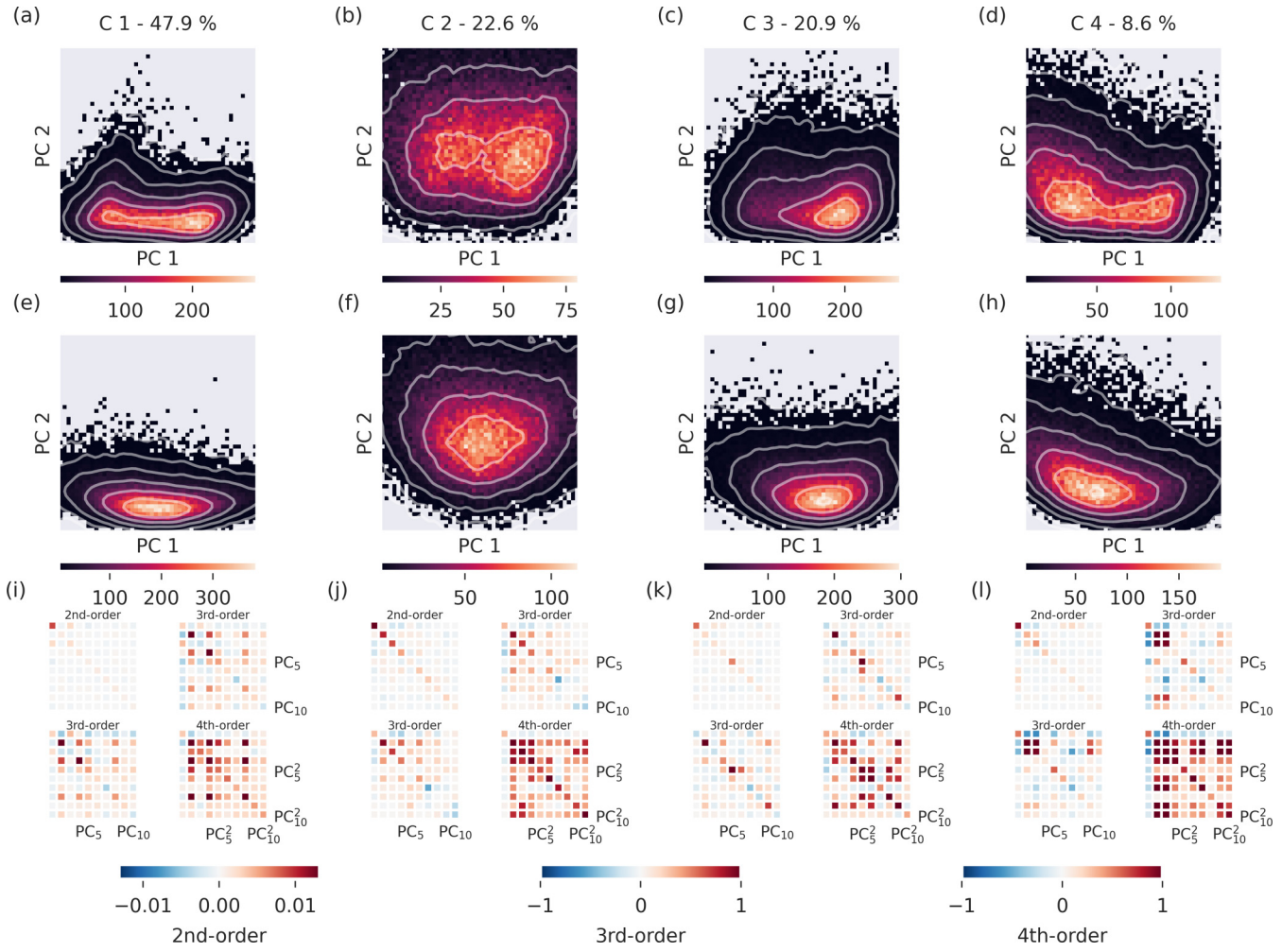


FIG. 15. Statistical characteristics of latent components in Neuropixels data reveal distinct non-Gaussian structures. Approximate marginalized distributions and higher-order statistics for each latent component. [(a)–(d)] Marginalized distributions of each latent component from Neuropixels data, projected onto the first two PCA directions. Light-gray contours indicate isolines of equal likelihood. Component weights show C1 as the dominant component (47.9%), followed by C2 (22.6%), C3 (20.9%), and C4 (8.6%). [(e)–(h)] Marginalized distributions when the inverse mapping of the network is approximated by a quadratic function [Eq. (10)]. [(i)–(l)] Second- to fourth-order cumulants computed from the quadratic approximation. The cumulant analysis reveals pronounced non-Gaussian statistics, with strong third- and 4thfourth-order correlations indicating significant departures from Gaussian behavior.

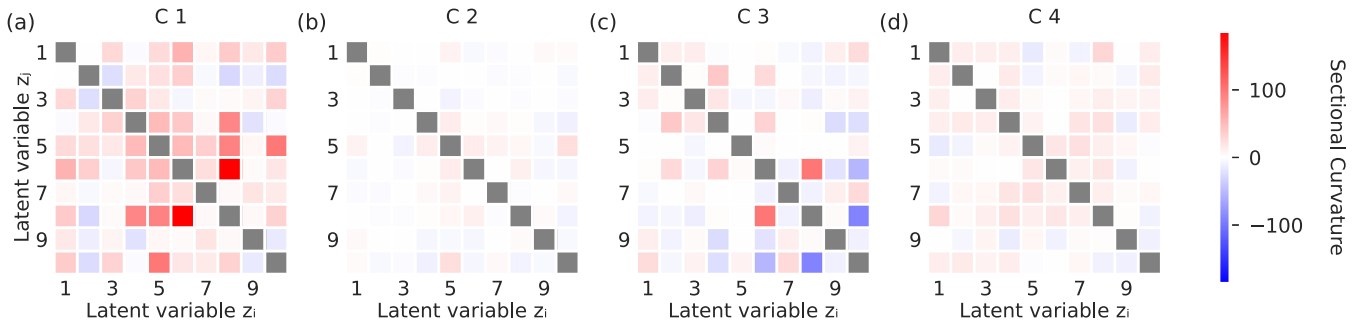


FIG. 16. Geometrical characteristics of latent components in Neuropixels data show large positive scalar curvatures. Panels (a)–(d) show the sectional curvature matrices for each of the four components of the latent distribution. These values are computed from the quadratically approximated inverse network. Red indicates positive curvature, while blue indicates negative curvature. Component C1 exhibits the largest positive scalar curvature, indicating the most strongly curved manifold geometry. Despite mixed positive and negative sectional curvatures visible in the matrices, the overall positive scalar curvatures suggest that the Neuropixels data lie on manifolds with predominantly elliptic (spherelike) rather than hyperbolic (saddle-point-like) geometry. This contrasts with the negative scalar curvatures observed in the macaque visual cortex data, suggesting different manifold structures across recording modalities.

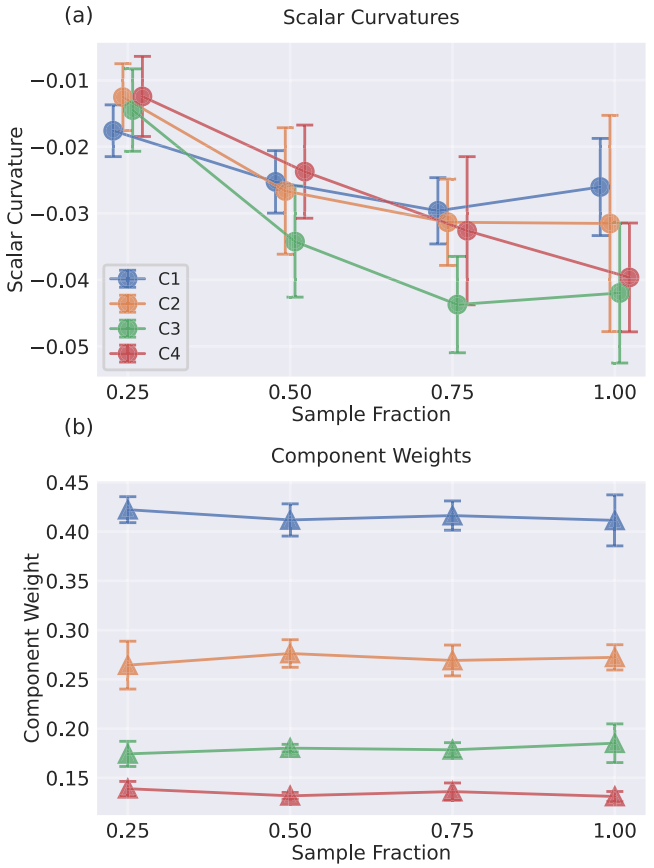


FIG. 17. Sample complexity analysis of geometric estimates. Dependency of estimated geometric quantities on sample fraction for session L_RS_250717. Analysis used 25%, 50%, 75%, and 100% of available data with 10 independent runs per fraction. Components are reordered by weight independently within each individual run (C1 = highest weight in that specific run, C2 = second highest, etc.). Points show means across the 10 runs; error bars indicate one standard deviation. (a) Scalar curvatures: While individual components show variable trajectories across sample fractions, the relative ordering and negative values remain consistent. (b) Component weights: High stability across all sample fractions confirms robust mixture decomposition, with consistent ranked hierarchy.

Thus, we get

$$\begin{aligned} Z(\boldsymbol{j}) &= e^{j^T c} \int d^n z \frac{1}{(2\pi)^{n/2}} \\ &\quad \times \left(e^{-\frac{1}{2}(z-\Sigma(j)B^T j)^T \Sigma^{-1}(j)(z-\Sigma(j)B^T j)} e^{\frac{1}{2} j^T B \Sigma(j) B^T j}\right) \\ &= e^{j^T c} e^{\frac{1}{2} j^T B \Sigma(j) B^T j} (\det(\Sigma(\boldsymbol{j})))^{1/2}. \end{aligned}$$

From here, the cumulant generating function directly follows as

$$\begin{aligned} W(\boldsymbol{j}) &= \ln Z(\boldsymbol{j}) \\ &= \boldsymbol{j}^T \boldsymbol{c} + \frac{1}{2} \boldsymbol{j}^T B \left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha\right)^{-1} B^T \boldsymbol{j} \\ &\quad - \frac{1}{2} \ln \det \left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha\right). \end{aligned}$$

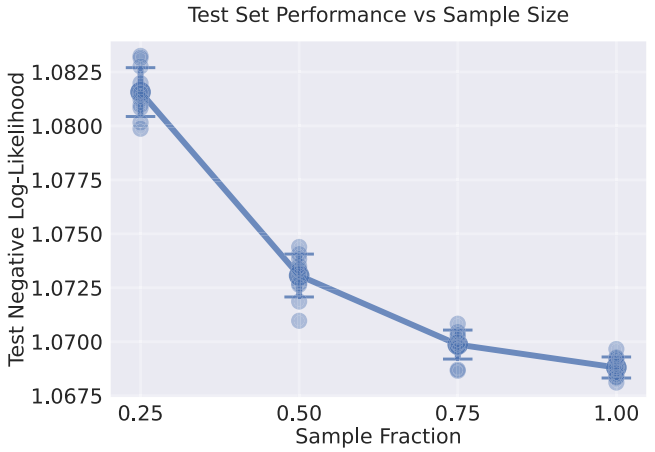


FIG. 18. Test set performance vs sample size. Generalization performance measured by test log-likelihood as a function of training sample fraction for session L_RS_250717. Models were trained on 25%, 50%, 75%, and 100% of available data with 10 independent runs per fraction. Each model was evaluated on its corresponding subsample test split (20% holdout). Points show mean test negative log-likelihood across runs; error bars indicate one standard deviation.

We expand the term $\Sigma(\boldsymbol{j})$ using the geometric series, given that we will be only interested in derivatives by j at $j = 0$,

$$\begin{aligned} \Sigma(\boldsymbol{j}) &= \left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha\right)^{-1} \\ &= \sum_{k=0}^{\infty} \left(\sum_\alpha A^\alpha j^\alpha\right)^k. \end{aligned}$$

To find a series expansion, we use that $\sum_\alpha A^\alpha j^\alpha$ is symmetric and therefore can be decomposed into its eigenvalues λ_l. It thus follows

$$\begin{aligned} \ln \det \left(\mathbb{I} - \sum_\alpha A^\alpha j^\alpha\right)\Big) &= \sum_l \ln(1-\lambda_l) \\ &= -\sum_l \sum_{k=0}^{\infty} \frac{\lambda_l^k}{k} \\ &= -\sum_{k=0}^{\infty} \frac{\sum_l \lambda_l^k}{k} \\ &= -\sum_{k=0}^{\infty} \frac{\mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha\right)^k\right)}{k}. \end{aligned}$$

We can now write the cumulant-generating function as an expansion in $\boldsymbol{j}$,

$$\begin{aligned} W(\boldsymbol{j}) &= \boldsymbol{j}^T \boldsymbol{c} + \frac{1}{2} \sum_{k=0}^{\infty} \boldsymbol{j}^T B \left(\sum_\alpha A^\alpha j^\alpha\right)^k B^T \boldsymbol{j} \\ &\quad + \frac{1}{2} \sum_{k=1}^{\infty} \frac{\mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha\right)^k\right)}{k} \end{aligned}$$

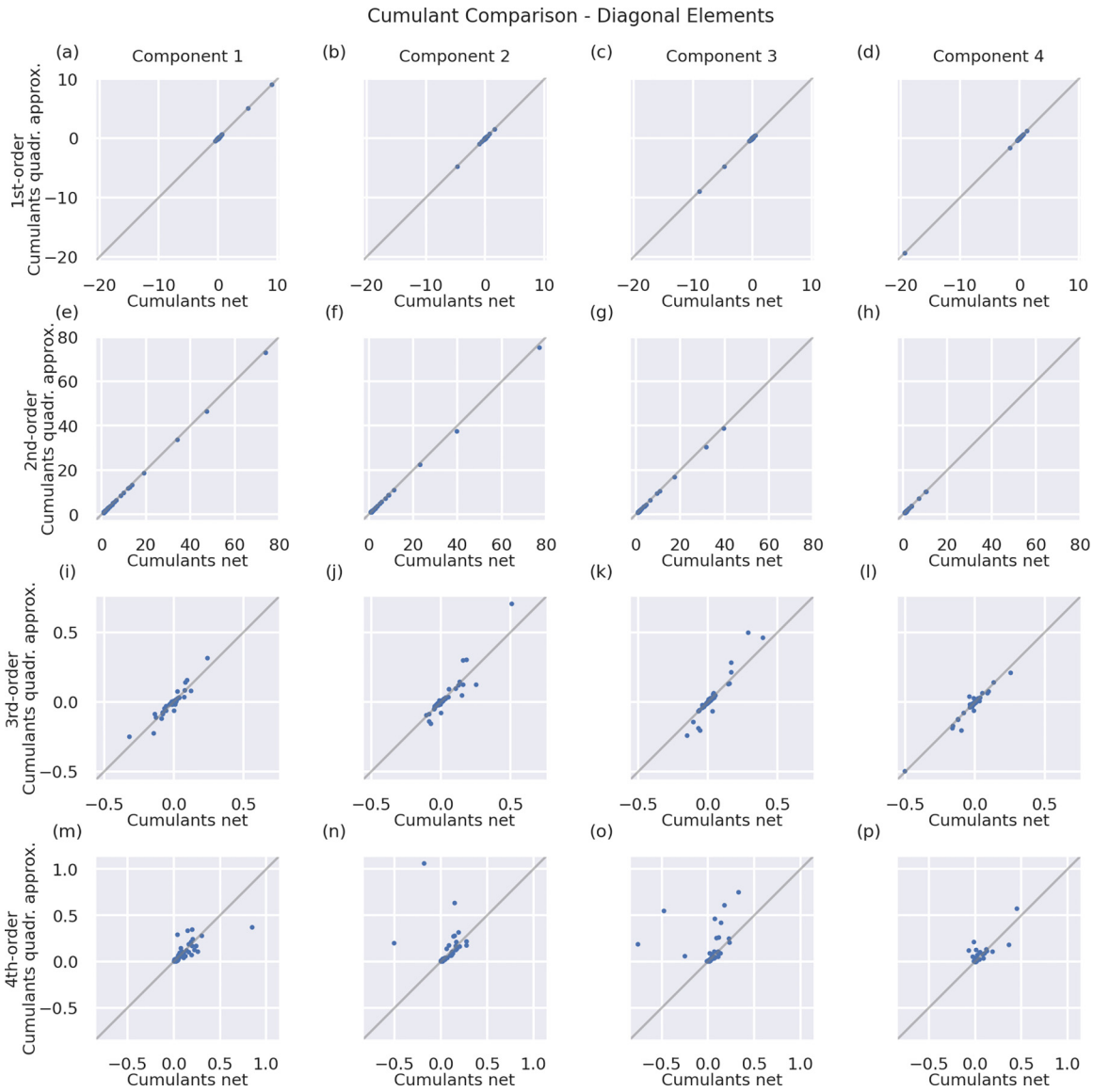


FIG. 19. Comparison of diagonal elements of cumulants between the original inverse mapping of the network f_θ^{-1} [Eq. (2)] and its quadratic approximation [Eq. (10)] for each of the four components. Each row corresponds to a different order of cumulants, from first order [panels (a)–(d)] to fourth order [panels (m)–(p)]. Each column corresponds to one of the four latent components.

$$= \sum_{k=1}^{\infty} \left[\delta_{k,1} \boldsymbol{j}^T \boldsymbol{c} + (1-\delta_{k,1}) \frac{1}{2} \boldsymbol{j}^T B \left(\sum_\alpha A^\alpha j^\alpha \right)^{k-2} B^T \boldsymbol{j} + \frac{1}{2} \frac{\mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha \right)^k \right)}{k} \right].$$

It is properly normalized as $W(\mathbf{0}) = 0$.

We can now read off the terms of different orders to obtain the cumulants, which we do here for four orders. At first order we have

$$W^1(\boldsymbol{j}) = \boldsymbol{j}^T \boldsymbol{c} + \frac{1}{2} \mathrm{Tr}\left(\sum_\alpha A^\alpha j^\alpha \right)$$
$$\langle\langle x^\alpha \rangle\rangle = c^\alpha + \frac{1}{2} \mathrm{Tr}(A^\alpha),$$

which yields the mean. At second order, we get the covariance

$$W^2(\boldsymbol{j}) = \frac{1}{2} \boldsymbol{j}^T B B^T \boldsymbol{j} + \frac{1}{4} \mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha \right)^2 \right)$$
$$\langle\langle x^\alpha x^\beta \rangle\rangle = [BB^T]^{\alpha\beta} + \frac{1}{4} \frac{\partial^2}{\partial j_\alpha \partial j_\beta} \mathrm{Tr}\left(\sum_{\alpha',\beta'} (A^{\alpha'} j^{\alpha'})(A^{\beta'} j^{\beta'}) \right)$$
$$= [BB^T]^{\alpha\beta} + \frac{1}{4} \sum_{(\alpha',\beta') \in \mathcal{P}(\alpha,\beta)} \mathrm{Tr}(A^{\alpha'} A^{\beta'}),$$

where $\mathcal{P}(\alpha, \ldots, \beta)$ is the set of all permutations of the given set of indices. At third order

$$W^3(\boldsymbol{j}) = \frac{1}{2} \boldsymbol{j}^T B \left(\sum_\alpha A^\alpha j^\alpha \right) B^T \boldsymbol{j} + \frac{1}{6} \mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha \right)^3 \right),$$

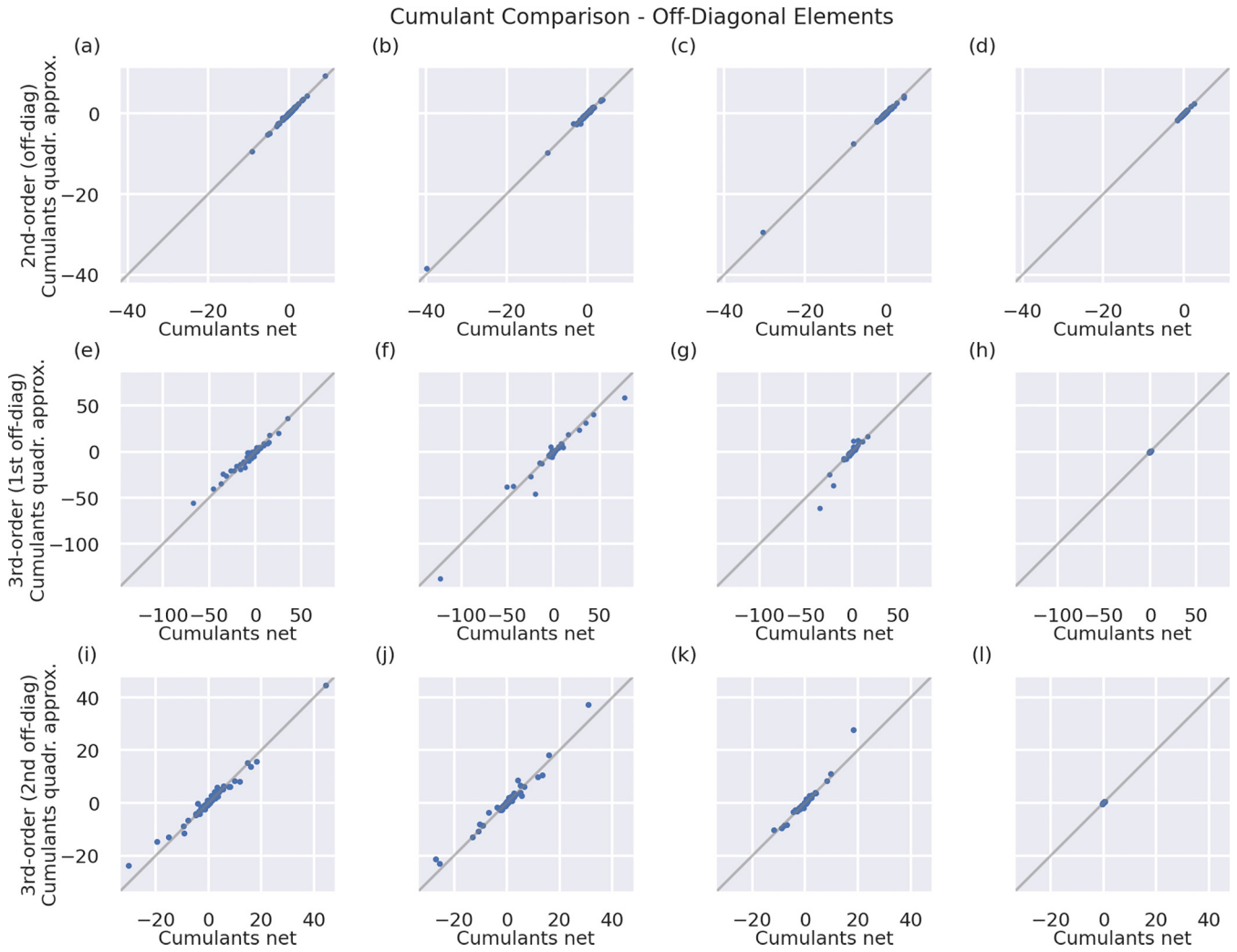


FIG. 20. Comparison of off-diagonal elements of cumulants between the original inverse mapping of the network f_θ^{-1} [Eq. (2)] and its quadratic approximation [Eq. (10)] for each of the four components. The first row shows off-diagonal elements for the second-order cumulants. The second and third rows show first- and second-order off-diagonal elements for third-order cumulants. Each column corresponds to one of the four latent components.

the third-order cumulants follow as

$$\langle\langle x^\alpha x^\beta x^\gamma\rangle\rangle = \frac{1}{2}\sum_{(\alpha',\beta',\gamma')\in\mathcal{P}(\alpha,\beta,\gamma)} [BA^{\beta'}B^T]^{\alpha'\gamma'} + \frac{1}{6}\sum_{(\alpha',\beta',\gamma')\in\mathcal{P}(\alpha,\beta,\gamma)} \mathrm{Tr}(A^{\alpha'}A^{\beta'}A^{\gamma'}).$$

At fourth order,

$$W^4(\boldsymbol{j}) = \frac{1}{2}\boldsymbol{j}^T B\left(\sum_\alpha A^\alpha j^\alpha\right)^2 B^T\boldsymbol{j} + \frac{1}{8}\mathrm{Tr}\left(\left(\sum_\alpha A^\alpha j^\alpha\right)^4\right),$$

we obtain the fourth-order cumulants as

$$\langle\langle x^\alpha x^\beta x^\gamma x^\delta\rangle\rangle = \frac{1}{2}\sum_{(\alpha',\beta',\gamma',\delta')\in\mathcal{P}(\alpha,\beta,\gamma,\delta)} [BA^{\alpha'}A^{\beta'}B^T]^{\gamma'\delta'} + \frac{1}{8}\sum_{(\alpha',\beta',\gamma',\delta')\in\mathcal{P}(\alpha,\beta,\gamma,\delta)} \mathrm{Tr}(A^{\alpha'}A^{\beta'}A^{\gamma'}A^{\delta'}).$$

In Fig. 8, we show results for the covariance $\langle\langle x^\alpha x^\beta\rangle\rangle$, the third-order cumulants $\langle\langle x^\alpha x^\beta x^\beta\rangle\rangle$, and the fourth-order cumulants $\langle\langle x^\alpha x^\alpha x^\beta x^\beta\rangle\rangle$.

APPENDIX H: CURVATURE

In this section, we derive the key geometric quantities that describe the curvature of data manifolds when mapped from a flat latent space. Specifically, we will focus on the Riemannian curvature tensor R^i_{klj}, the sectional curvature $K(e_i, e_j)$ and the scalar curvature R.

We start by considering the mapping of latent variables z_j to data space x^α through a quadratic function given by Eq. (10). For simplicity, we adopt the Einstein summation convention throughout this derivation. Greek letters denote indices in data space (upper indices), while Latin letters refer to indices in latent space (lower indices).

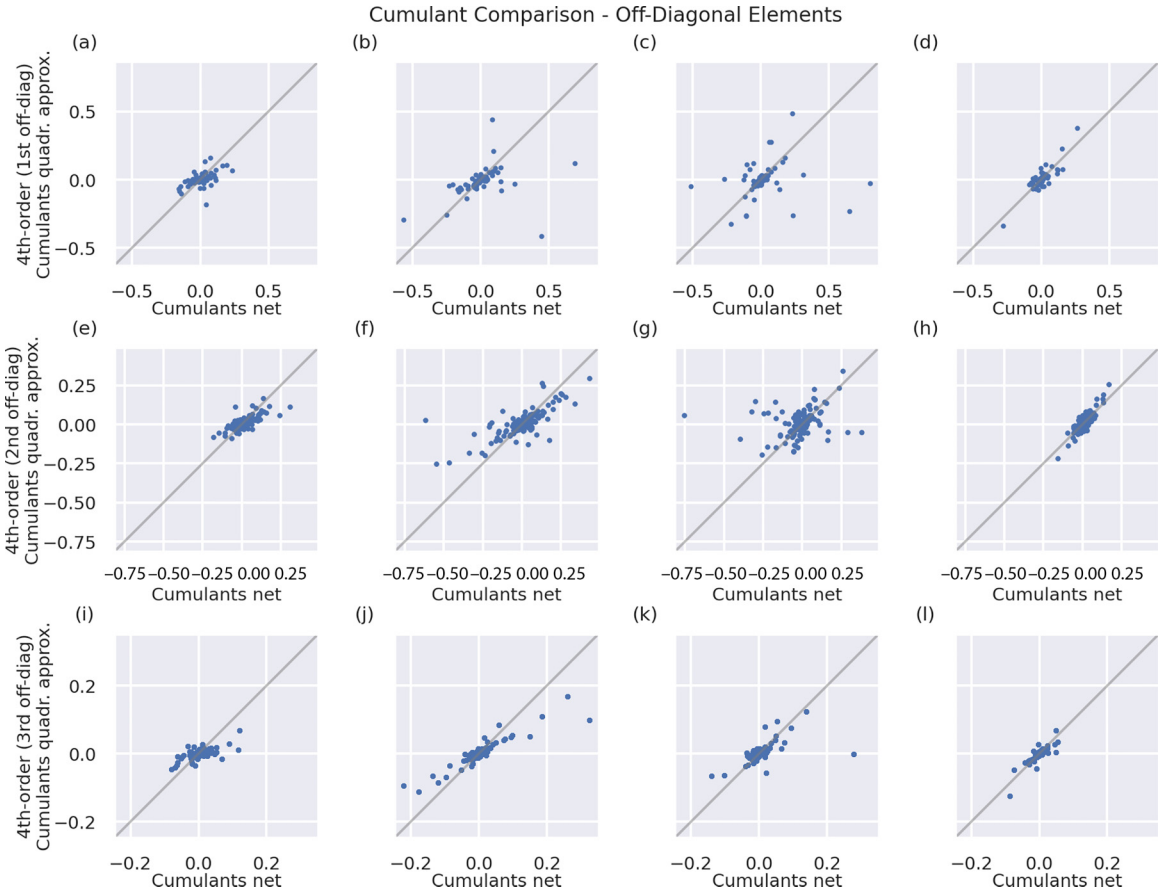


FIG. 21. Comparison of off-diagonal elements of cumulants between the original inverse mapping of the network f_θ^{-1} [Eq. (2)] and its quadratic approximation [Eq. (10)] across four components for fourth-order cumulants. Each row corresponds to a different degree of off-diagonality of cumulant, from first-order off-diagonal [panels (a)–(d)] to third-order off-diagonal [panel (i)–(l)], demonstrating the effectiveness of the quadratic fit.

1. Step 1: Tangent vectors

To understand how latent variables act as coordinates of the data space, we first compute tangent vectors (see chap. 3 in Ref. [92]). These vectors represent partial derivatives of data coordinates with respect to latent variables:

$$e_i^\alpha = \frac{\partial x^\alpha}{\partial z_i} = B_i^\alpha + A_{ij}^\alpha z_j. \tag{H1}$$

This equation shows how infinitesimal changes in each latent variable influence the position within the manifold.

2. Step 2: Metric tensor

The metric tensor encodes local geometric information such as distances and angles between directions on the manifold. It is constructed using the tangent vectors (see chap. 2 in Ref. [93]):

$$g_{ij} = e_i^\alpha e_j^\alpha.$$

Expressed in terms of the tangent vectors [Eq. (H1)] the metric tensor is

$$\begin{aligned} g_{ij} &= \left(B_i^\alpha + A_{ik}^\alpha z_k\right)\left(B_j^\alpha + A_{jl}^\alpha z_l\right) \\ &= \underbrace{B_i^\alpha B_j^\alpha}_{:=\gamma_{ij}} + \underbrace{\left(B_i^\alpha A_{jk}^\alpha + B_j^\alpha A_{ik}^\alpha\right)}_{:=w_{ij,k}} z_k + \underbrace{A_{ik}^\alpha A_{jl}^\alpha}_{:=h_{ik,jl}} z_k z_l \\ &= \gamma_{ij} + w_{ij,k} z_k + h_{ik,jl} z_k z_l, \end{aligned}$$

where we introduce γ_{ij}, $w_{ij,k}$, and $h_{ik,jl}$ as the coefficients for the different powers of z of the metric tensor.

For the derivation of the Riemannian curvature, we will also need the partial derivatives of the metric tensor:

$$\begin{aligned} g_{ij,k} &= \frac{\partial}{\partial z_k} g_{ij} \\ &= \frac{\partial}{\partial z_k}(\gamma_{ij} + w_{ij,m} z_{km} + h_{ij,ml} z_m z_l) \\ &= w_{ij,k} + (h_{ik,jl} + h_{il,jk}) z_l. \end{aligned}$$

Note on Taylor expansion around $z = 0$

To simplify our calculations, we utilize a Taylor expansion around $z = 0$. This approach is chosen without loss of generality because the expansion could equivalently be performed around any point $z = z^*$. By focusing on $z = 0$, we can derive expressions for the curvature tensors that are valid at this reference point. This expression can then be generalized to other points to obtain $R^i_{jkl}(z = z^*)$.

3. Step 3: Inverse metric tensor

Next we determine the inverse metric tensor (where we also use upper indices) by expanding it around $z = 0$:

$$g^{ij} = g^{ij}_{(0)} + g^{ij}_{(1),k} z_k + g^{ij}_{(2),kl} z_k z_l + \mathcal{O}(z^3). \tag{H2}$$

These components are calculated by ensuring that their product with the metric tensor g_{jm} equals the identity matrix (see chap. 8 in Ref. [92]):

$$\begin{aligned} g^{ij} g_{jm} &= \delta_{im} \\ &= \left(g^{ij}_{(0)} + g^{ij}_{(1),k} z_k + g^{ij}_{(2),kl} z_k z_l + \mathcal{O}(z^3)\right) \\ &\quad \times \left(\gamma_{jm} + w_{jm,k} z_k + h_{jk,ml} z_k z_l\right) \\ &= g^{ij}_{(0)} \gamma_{jm} + \left(g^{ij}_{(0)} w_{jm,k} + g^{ij}_{(1),k} \gamma_{jm}\right) z_k \\ &\quad + \left(g^{ij}_{(0)} h_{jk,ml} + g^{ij}_{(1),k} w_{jm,l} + g^{ij}_{(2),kl} \gamma_{jm}\right) z_k z_l \\ &\quad + \mathcal{O}(z^3). \end{aligned}$$

Since the inverse relationship must hold for all z and the polynomial of z of different powers are independent, the coefficients in front of each power of z must vanish separately. We can hence read of the terms up to quadratic order. At zeroth order in z we have

$$g^{ij}_{(0)} = (\gamma^{-1})_{ij} = \gamma^{ij}. \tag{H3}$$

The first-order term $\propto z$ yields

$$\begin{aligned} 0 &= g^{ij}_{(0)} w_{jm,k} + g^{ij}_{(1),k} \gamma_{jm} \\ g^{ij}_{(1),k} &= -g^{in}_{(0)} w_{nm,k} g^{mj}_{(0)} = -\gamma^{in} w_{nm,k} \gamma^{mj}, \end{aligned} \tag{H4}$$

where we used Eq. (H3). The second order yields

$$\begin{aligned} 0 &= g^{ij}_{(0)} h_{jk,ml} + g^{ij}_{(1),k} w_{jm,l} + g^{ij}_{(2),kl} \gamma_{jm} \\ g^{ij}_{(2),kl} &= -g^{in}_{(0)} h_{nk,ml} g^{mj}_{(0)} - g^{in}_{(1),k} w_{nm,l} g^{mj}_{(0)} \\ &\overset{(H3\&H4)}{=} -\gamma^{in} h_{nk,ml} \gamma^{mj} + \gamma^{io} w_{om,k} \gamma^{mn} w_{nm,l} \gamma^{mj}. \end{aligned}$$

4. Step 4: Christoffel symbols

Christoffel symbols are essential for computing curvature because they capture connections between tangent spaces (see chap. 5 in Ref. [93]). We expand them up to second order in z:

$$\begin{aligned} \Gamma^i_{kl} &= \tfrac{1}{2} g^{im} (g_{mk,l} + g_{ml,k} - g_{kl,m}) \\ &= \tfrac{1}{2} g^{im} (w_{mk,l} + w_{ml,k} - w_{kl,m}) \\ &\quad + \tfrac{1}{2} g^{im} ((h_{ml,kn} + h_{mn,kl} + h_{mk,ln} + h_{mn,lk}) z_n) \\ &\quad - \tfrac{1}{2} g^{im} ((h_{km,ln} + h_{kn,lm}) z_n) \\ &= \tfrac{1}{2} g^{im} (w_{mk,l} + w_{ml,k} - w_{kl,m}) + g^{im} h_{mn,kl} z_n \\ &\overset{(H2)}{=} \tfrac{1}{2} (\gamma^{im} - \gamma^{in} w_{no,a} \gamma^{om} z_a + \mathcal{O}(z^2)) \\ &\quad \times (w_{mk,l} + w_{ml,k} - w_{kl,m}) \\ &\quad + (\gamma^{im} + \mathcal{O}(z)) h_{mn,kl} z_n \\ &= (\gamma^{im} - \gamma^{in} w_{no,a} \gamma^{om} z_a) B^\alpha_m A^\alpha_{kl} \\ &\quad + \gamma^{im} h_{mn,kl} z_n + \mathcal{O}(z^2). \end{aligned}$$

For the last step, we used that

$$\begin{aligned} w_{mk,l} + w_{ml,k} - w_{kl,m} &= B^\alpha_m A^\alpha_{kl} + B^\alpha_k A^\alpha_{ml} + B^\alpha_m A^\alpha_{lk} \\ &\quad + B^\alpha_l A^\alpha_{mk} - B^\alpha_k A^\alpha_{lm} - B^\alpha_l A^\alpha_{km}. \\ &= 2 B^\alpha_m A^\alpha_{kl}. \end{aligned}$$

The partial derivative of the Christoffel symbols are also required, which up to linear order are as follows:

$$\frac{\partial \Gamma^i_{lj}}{\partial z^k} = -\gamma^{in} w_{no,k} \gamma^{om} B^\alpha_m A^\alpha_{lj} + \gamma^{im} h_{mk,lj} + \mathcal{O}(z).$$

5. Step 5: Riemannian curvature tensor

With all necessary components, we can now compute the Riemannian curvature tensor which quantifies intrinsic bending or warping at a point on manifold (see chap. 7 in Ref. [93]):

$$R^i_{klj} = \frac{\partial \Gamma^i_{lj}}{\partial z^k} - \frac{\partial \Gamma^i_{kj}}{\partial z^l} + \left(\Gamma^i_{kp} \Gamma^p_{lj} - \Gamma^i_{lp} \Gamma^p_{kj}\right).$$

We perform the different parts of this calculation

$$\begin{aligned} \frac{\partial \Gamma^i_{lj}}{\partial z^k} - \frac{\partial \Gamma^i_{kj}}{\partial z^l} &= -\gamma^{in} w_{no,k} \gamma^{om} B^\alpha_m A^\alpha_{lj} \\ &\quad + \gamma^{in} w_{no,l} \gamma^{om} B^\alpha_m A^\alpha_{kj} \\ &\quad + \gamma^{im} h_{mk,lj} - \gamma^{im} h_{ml,kj} + \mathcal{O}(z) \\ &= \gamma^{in} B^\alpha_m \gamma^{om} B^\beta_n \left(-A^\beta_{ok} A^\alpha_{lj} + A^\beta_{ol} A^\alpha_{kj}\right) \\ &\quad + \gamma^{in} B^\alpha_m \gamma^{om} B^\beta_o \left(-A^\beta_{nk} A^\alpha_{lj} + A^\beta_{nl} A^\alpha_{kj}\right) \\ &\quad + \gamma^{im} h_{mk,lj} - \gamma^{im} h_{ml,kj} + \mathcal{O}(z). \end{aligned}$$

The product of the Christoffel symbols becomes

$$\Gamma^i_{kp} \Gamma^p_{lj} = \gamma^{im} B^\alpha_m A^\alpha_{kp} \gamma^{pn} B^\beta_n A^\beta_{lj} + \mathcal{O}(z).$$

Further, the substraction of products of Christoffel symbols is as follows:

$$\begin{aligned} \Gamma^i_{kp} \Gamma^p_{lj} - \Gamma^i_{lp} \Gamma^p_{kj} &= \gamma^{im} B^\alpha_m A^\alpha_{kp} \gamma^{pn} B^\beta_n A^\beta_{lj} \\ &\quad - \gamma^{im} B^\alpha_m A^\alpha_{lp} \gamma^{pn} B^\beta_n A^\beta_{kj} + \mathcal{O}(z) \\ &= \gamma^{im} B^\alpha_m \gamma^{pn} B^\beta_n \left(A^\alpha_{kp} A^\beta_{lj} - A^\alpha_{lp} A^\beta_{kj}\right) + \mathcal{O}(z). \end{aligned}$$

The Riemannian curvature tensor follows thus as follows:

$$\begin{aligned} R^i_{klj} &= \frac{\partial \Gamma^i_{lj}}{\partial z^k} - \frac{\partial \Gamma^i_{kj}}{\partial z^l} + \left(\Gamma^i_{kp} \Gamma^p_{lj} - \Gamma^i_{lp} \Gamma^p_{kj}\right) \\ &= \gamma^{in} B^\alpha_m \gamma^{pm} B^\beta_n \left(-A^\beta_{pk} A^\alpha_{lj} + A^\beta_{pl} A^\alpha_{kj}\right) \\ &\quad + \gamma^{in} B^\alpha_m \gamma^{pm} B^\beta_p \left(-A^\beta_{nk} A^\alpha_{lj} + A^\beta_{nl} A^\alpha_{kj}\right) \\ &\quad + \gamma^{im} \left(A^\alpha_{mk} A^\alpha_{lj} - A^\alpha_{ml} A^\alpha_{kj}\right) \\ &\quad + \gamma^{in} B^\beta_m \gamma^{pm} B^\alpha_n \left(A^\alpha_{kp} A^\beta_{lj} - A^\alpha_{lp} A^\beta_{kj}\right) + \mathcal{O}(z) \\ &= \gamma^{in} B^\alpha_m \gamma^{pm} B^\beta_p \left(-A^\beta_{nk} A^\alpha_{lj} + A^\beta_{nl} A^\alpha_{kj}\right) \\ &\quad + \gamma^{im} \left(A^\alpha_{mk} A^\alpha_{lj} - A^\alpha_{ml} A^\alpha_{kj}\right) + \mathcal{O}(z) \end{aligned}$$

$$
\begin{aligned}
&= \gamma^{im} B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} \big(- A^{\beta}_{mk} A^{\alpha}_{lj} + A^{\beta}_{ml} A^{\alpha}_{kj} \big) \\
&\quad + \gamma^{im} \big(A^{\alpha}_{mk} A^{\alpha}_{lj} - A^{\alpha}_{ml} A^{\alpha}_{kj} \big) + \mathcal{O}(z) \\
&= \gamma^{im} A^{\alpha}_{lj} \big(A^{\alpha}_{mk} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} A^{\beta}_{mk} \big) \\
&\quad - \gamma^{im} A^{\alpha}_{kj} \big(A^{\alpha}_{ml} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} A^{\beta}_{ml} \big) + \mathcal{O}(z) \\
&= \gamma^{im} A^{\alpha}_{jl} \big(\delta_{\alpha\beta} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} \big) A^{\beta}_{mk} \\
&\quad - \gamma^{im} A^{\alpha}_{kj} \big(\delta_{\alpha\beta} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} \big) A^{\beta}_{ml} + \mathcal{O}(z) \\
&= \gamma^{im} \big(\delta_{\alpha\beta} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} \big) \big(A^{\alpha}_{jl} A^{\beta}_{mk} - A^{\alpha}_{kj} A^{\beta}_{ml} \big) \\
&\quad + \mathcal{O}(z).
\end{aligned}
$$

For $z = 0$, it follows that

$$
\begin{aligned}
&R^{i}_{klj}(z=0) \\
&\quad = \gamma^{im} \big(\delta_{\alpha\beta} - B^{\alpha}_{n} \gamma^{pn} B^{\beta}_{p} \big) \big(A^{\alpha}_{jl} A^{\beta}_{mk} - A^{\alpha}_{kj} A^{\beta}_{ml} \big) \\
&\quad = g^{im}(z=0) \big(\delta_{\alpha\beta} - e^{\alpha}_{n}(z=0) g^{pn}(z=0) e^{\beta}_{p}(z=0) \big) \times \\
&\qquad \times \big(A^{\alpha}_{jl} A^{\beta}_{mk} - A^{\alpha}_{kj} A^{\beta}_{ml} \big).
\end{aligned}
$$

Given that we performed the calculations here around $z = 0$ without loss of generality, we can conclude for all z that

$$
\begin{aligned}
R^{i}_{klj}(z) = g^{im}(z) \underbrace{\big(\delta_{\alpha\beta} - e^{\alpha}_{n}(z) g^{pn}(z) e^{\beta}_{p}(z) \big)}_{:= P^{\alpha\beta}} \\
\times \big(A^{\alpha}_{jl} A^{\beta}_{mk} - A^{\alpha}_{kj} A^{\beta}_{ml} \big).
\end{aligned}
$$

Note that there is no z dependency for the second derivative of Eq. (10).

Further, we make use of the fact that $P^{\alpha\beta} = P^{\alpha\gamma} P^{\gamma\beta}$, to introduce

$$
H^{\gamma}_{ij} := P^{\alpha\gamma} A^{\alpha}_{jl}.
$$

With this definition, we can write the curvature tensor as

$$
R^{i}_{klj} = g^{im} \big(H^{\gamma}_{jl} H^{\gamma}_{mk} - H^{\gamma}_{kj} H^{\gamma}_{ml} \big).
$$

Like this we recover the description used in Ref. [99].

After simplifying terms using the projection operator $P^{\alpha\beta}$, it becomes

$$
R^{i}_{klj} = g^{im} P^{\alpha\beta} \big(A^{\alpha}_{jl} A^{\beta}_{mk} - A^{\alpha}_{kj} A^{\beta}_{ml} \big). \tag{H5}
$$

This expression captures interactions between different dimensions contributing towards the overall shape distortion.

6. Step 6: Sectional curvature

Sectional curvature measures how much the manifold bends along specific two-dimensional planes spanned by pairs of tangent vectors (see chap. 7 in Ref. [93]):

$$
\begin{aligned}
K(e_i, e_j) &= \frac{\langle R(e_i, e_j) e_i, e_j \rangle}{\langle e_i, e_i \rangle \langle e_j, e_j \rangle - \langle e_i, e_j \rangle^2} \\
&= \frac{g_{jm} R^{m}_{jii}}{g_{ii} g_{jj} - g^2_{ij}}.
\end{aligned}
$$

We enter Eq. (H5) into this formula:

$$
K(e_i, e_j) = \frac{P^{\alpha\beta} \big(A^{\alpha}_{ii} A^{\beta}_{jj} - A^{\alpha}_{ij} A^{\beta}_{ij} \big)}{g_{ii} g_{jj} - g^2_{ij}}.
$$

7. Step 7: Scalar curvature

Finally calculating the scalar curvature provides an aggregate measure summarizing total intrinsic curvature occurring throughout any direction at a given location on the manifold.

To calculate this, we first determine the Ricci curvature (see chap. 7 in Ref. [93]):

$$
\begin{aligned}
R_{jl} &= R^{c}_{cjl} \\
&= g^{cm} P^{\alpha\beta} \big(A^{\alpha}_{jl} A^{\beta}_{mc} - A^{\alpha}_{cj} A^{\beta}_{ml} \big).
\end{aligned}
$$

Finally, we arrive at the scalar curvature (see chap. 7 in Ref. [93]):

$$
\begin{aligned}
R &= g^{jl} R_{jl} \\
&= g^{jl} g^{cm} P^{\alpha\beta} \big(A^{\alpha}_{jl} A^{\beta}_{mc} - A^{\alpha}_{cj} A^{\beta}_{ml} \big).
\end{aligned}
$$

APPENDIX I: SEPARATION OF CONTRIBUTIONS TO NON-GAUSSIAN CORRELATIONS FROM CURVATURE AND NON-GAUSSIANITY OF THE LATENT SPACE

The quadratic map may be useful to distinguish the two origins of non-Gaussian correlations through the measure of curvature, which will vanish on a flat manifold. More precisely, in the case that data lie on a flat manifold, the map between the data space x and latent space z is one in which any local basis at different root points spans the same space

$$
\mathcal{T} \equiv \mathcal{T}_z = \mathrm{span}(e_1(z), \ldots, e_N(z)) \quad \forall z, \tag{I1}
$$

where $e^{\alpha}_{i}(z) = \frac{\partial x_\alpha}{\partial z_i}(z)$ are the basis vectors that span the tangent space. The map $x(z)$ may still stretch, shrink, or rotate the tangent vectors within this span as a function of the root point z, but for all z the $e_i(z)$ lie in the same subspace. Stretching, shrinking and rotation will still yield non-Gaussian correlations in the data space x. The basis vectors found by a method such as PCA, which assumes Gaussian distributed data, will yield the same space and these basis vectors can then be used to describe the non-Gaussian distribution within it. A special case of this situation is an affine-linear map

$$
x^{\alpha} = B^{\alpha}_{i} z_i + c^{\alpha},
$$

where $e^{\alpha}_{i}(z) = B^{\alpha}_{i}$, so that they are in particular independent of the root point and thus (I1) is true.

The curvature we compute from the quadratic approximation

$$
x^{\alpha} = \frac{1}{2} \sum_{i,j} A^{\alpha}_{ij} z_i z_j + \sum_{i} B^{\alpha}_{i} z_i + c^{\alpha} \tag{I2}
$$

can precisely indicate if non-Gaussian statistics arise due to the manifold being curved: The numerator of the sectional curvature (15)

$$
\sum_{\alpha,\beta} P^{\alpha\beta} \big(A^{\alpha}_{ii} A^{\beta}_{jj} - A^{\alpha}_{ij} A^{\beta}_{ij} \big) \tag{I3}
$$

measures for each pair of coordinates (i, j) how changing one of the coordinates i makes the other coordinates' local basis vector e_j leave the tangent space $\mathrm{span}(e_1(z), \ldots, e_N(z))$: Here the second derivatives appear, because one investigates how the basis vectors change $\frac{\partial}{\partial z_i} e^{\alpha}_{j}(z) = A^{\alpha}_{ij}$ and P is

the projection operator onto the orthogonal complement of $\mathrm{span}(e_1(z), \ldots, e_N(z))$.

In cases in which the curvature vanishes for all pairs of coordinates (i, j), the remaining cumulants can thus be attributed to non-Gaussian distributions of the latent space. The examples in Fig. 1(a) and 1(b) exactly display these cases. While the curvature for Fig. 1(a) is zero, the curvature for Fig. 1(b) is shown with a color code in Fig. 1(d). Both distributions display nonvanishing fourth-order correlations (a more asymmetric distribution might also include odd orders), but the curvature of the distribution in Fig. 1(a) vanishes while the curvature of the data in Fig. 1(b) does not, indicating that these data not only have interesting higher-order statistics but also lie on a curved manifold.

If one wants to estimate the cumulants that are caused by the deformation of the latent space alone, then one may approximate the quadratic mapping by one nonlinear mapping $\varphi : \mathbb{R}^d \mapsto \mathbb{R}^d$ within the d-dimensional latent space and another linear mapping $\psi : \mathbb{R}^d \mapsto \mathbb{R}^N$ from the latent to the data space as $x(z) = [\psi \circ \varphi](z)$. The resulting mapping then leads to vanishing curvature, but for general nonlinear φ may cause non-Gaussian correlations by deforming the latent space. Under this condition, the third-order tensor A^α_{ij} in (I2) can thus be decomposed as $A^\alpha_{ij} = \sum_{k=1}^{d} a^\alpha_k W^k_{ij}$. The curvature (I3) would then vanish, because all $\frac{\partial}{\partial z_i} e^\alpha_j(z) = A^\alpha_{ij}$ would lie within the d-dimensional subspace spanned by the a^α_k, so that the projection to the orthogonal complement vanishes.

APPENDIX J: SUPPLEMENTARY AND TABLES

In the last section, we show that our results are consistent across recording sessions. In order to show that, we present the results obtained for session L_RS_250717 also for the two sessions L_RS_090817 and L_RS_100817. Higher-order correlations in the neuronal data for these two sessions are shown in Figs. 22 and 23. The effect of the number of latent components on the approximation capabilities is shown in Figs. 24 and 25, respectively. The correspondence between latent mixture components and behavioral states is shown in Figs. 26 and 27. Statistical characteristics of the latent components for the two sessions are shown in Figs. 28 and 29. Geometrical characteristics of the latent components are shown in Figs. 30 and 31.

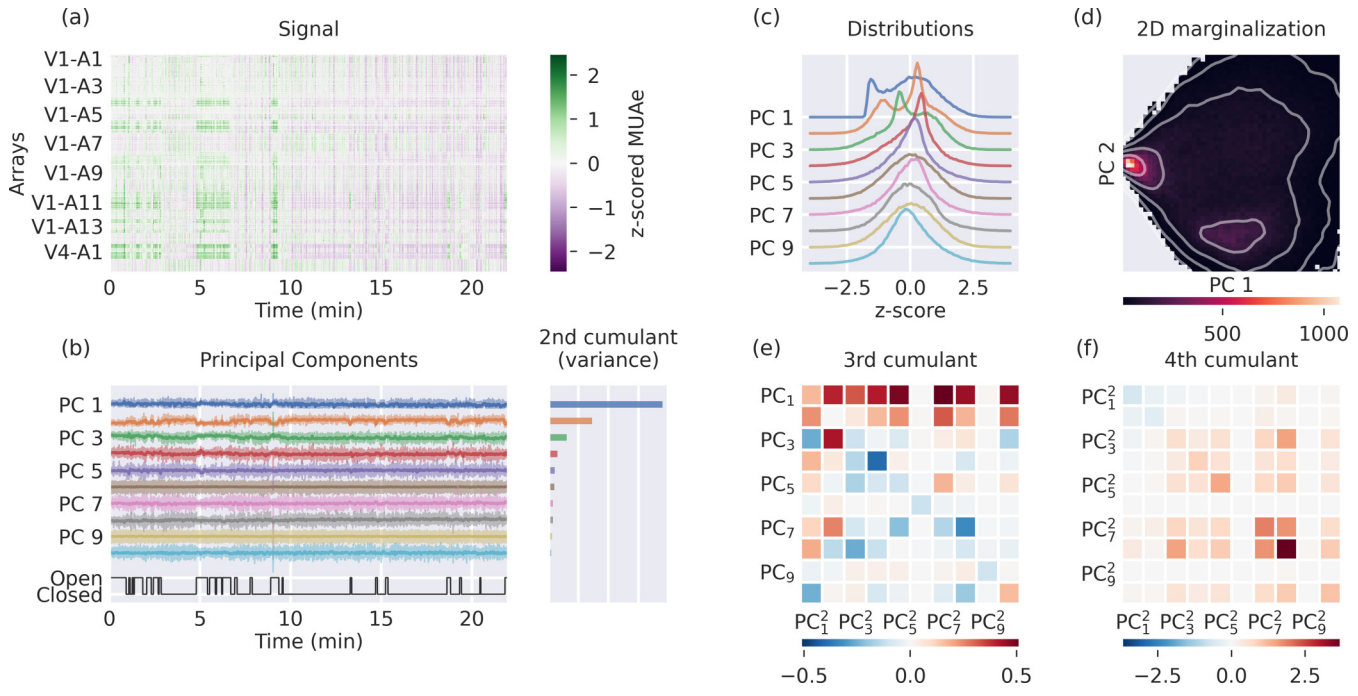


FIG. 22. Neuronal data exhibit higher-order correlations. (a) The original signal from one recording session (L_RS_090817), sampled at 100 Hz over 22.0 min. The signal is a MUAe containing 774 channels from 16 recording electrode arrays (14 in V1 and 2 in V4). Each channel is independently z scored (zero mean, unit variance) to ensure consistent scaling across neurons. (b) Time evolution of the first 10 PCs of the z-scored data. Transparent curves represent raw PCs, while solid curves show temporal smoothing using a Gaussian kernel with 1 s width. The labels of the behavioral state (eyes open or eyes closed) are shown below. The rapid decay of the eigenvalues σ_i^2 of the covariance matrix (second-order cumulants; on the right), corresponding to the shown PCs, is an indicator that the neural data lie on a lower-dimensional manifold. (c) Marginal distribution of the first 10 PCs. While some appear Gaussian, others clearly deviate from Gaussianity. (d) Two-dimensional marginalization along the first and second PCs. The color bar indicates the density of data points in the 2D histogram. [(e) and (f)] Third- and fourth-order cumulant. For both cases, the signals along the PCs are first divided by their standard deviation σ_i to normalize for the amplitude. Thus, panel (e) portrays x_i/σ_i against x_j^2/σ_j^2 and panel (f) x_i^2/σ_i^2 against x_j^2/σ_j^2. Color bars indicate cumulant magnitudes.

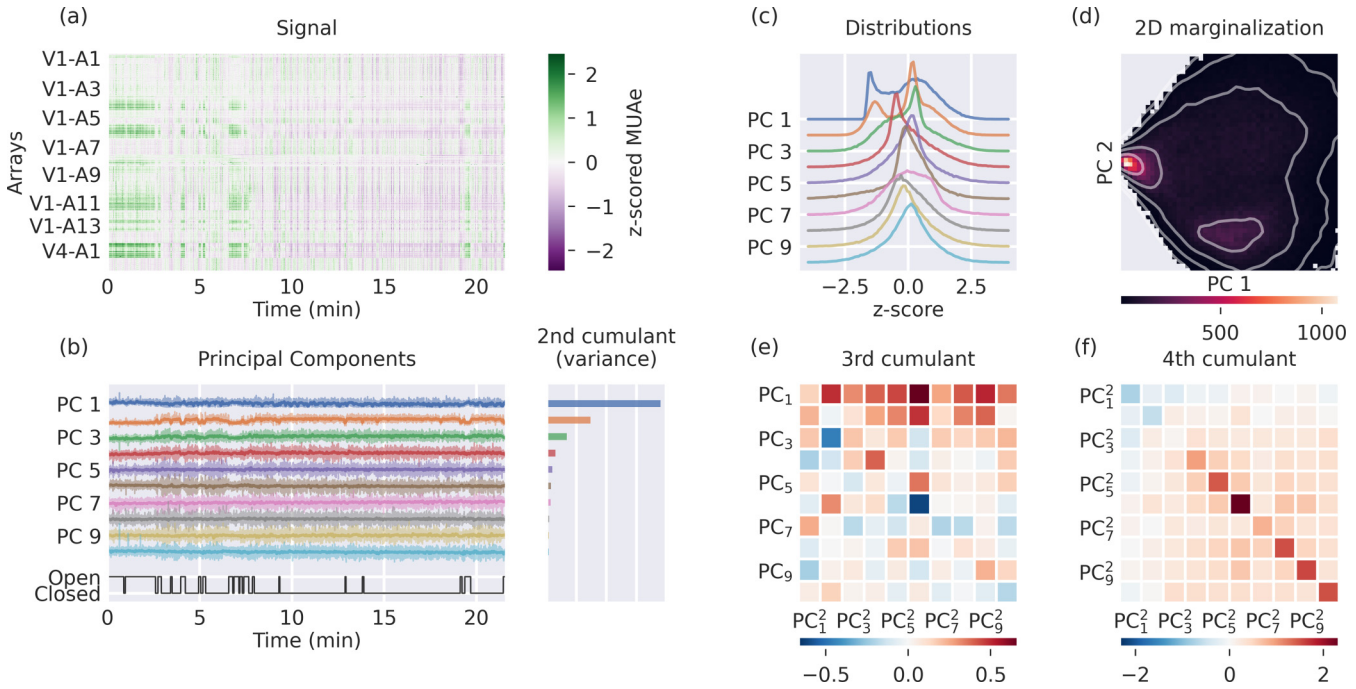


FIG. 23. Neuronal data exhibit higher-order correlations. (a) The original signal from one recording session (L_RS_100817), sampled at 100 Hz over 21.6 min. The signal is a MUAe containing 838 channels from 16 recording electrode arrays (14 in V1 and 2 in V4). Each channel is independently z scored (zero mean, unit variance) to ensure consistent scaling across neurons. (b) Time evolution of the first 10 PCs of the z-scored data. Transparent curves represent raw PCs, while solid curves show temporal smoothing using a Gaussian kernel with 1 s width. The labels of the behavioral state (eyes open or eyes closed) are shown below. The rapid decay of the eigenvalues σ_i^2 of the covariance matrix (second-order cumulants; on the right), corresponding to the shown PCs, is an indicator that the neural data lie on a lower-dimensional manifold. (c) Marginal distribution of the first 10 PCs. While some appear Gaussian, others clearly deviate from Gaussianity. (d) Two-dimensional marginalization along the first and second PCs. The color bar indicates the density of data points in the 2D histogram. [(e) and (f)] Third- and fourth-order cumulant. For both cases, the signals along the PCs are first divided by their standard deviation σ_i to normalize for the amplitude. Thus, panel (e) portrays x_i/σ_i against x_j^2/σ_j^2 and panel (f) x_i^2/σ_i^2 against x_j^2/σ_j^2. Color bars indicate cumulant magnitudes.

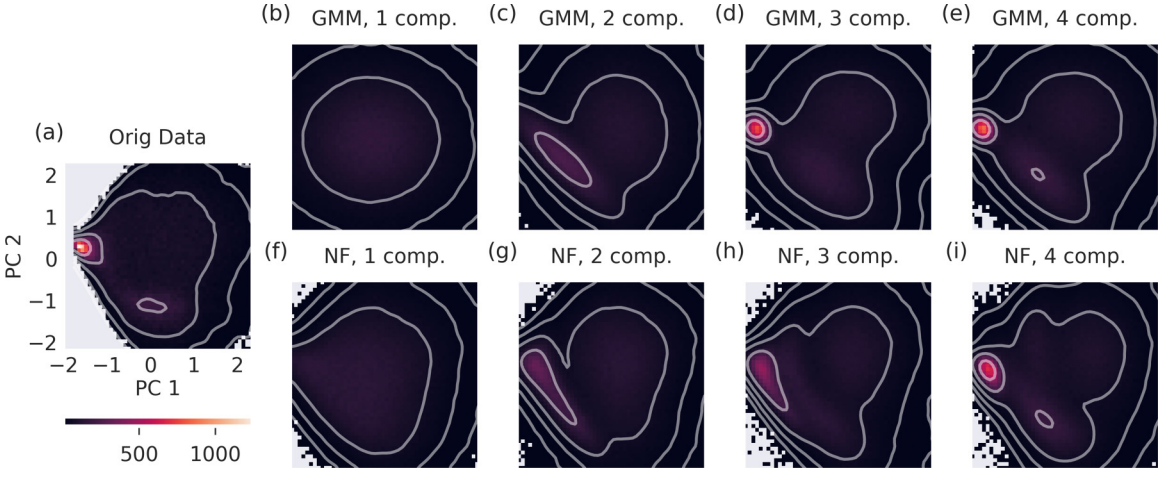


FIG. 24. Effect of number of latent components on the approximation capabilities: Marginalized distributions for the experimental data and learned models. (a) Marginalized distribution of one session of experimental data (L_RS_090817) along the first and second PCA directions. The color bar indicates the density of data points in the 2D histogram and applies to all panels. Light-gray isolines represent equal likelihood. [(b)–(e)] Marginalized probability densities for data sampled from GMMs with increasing numbers of latent components (one to four). [(f)–(i)] Corresponding distributions for NFs, also with one to four latent components.

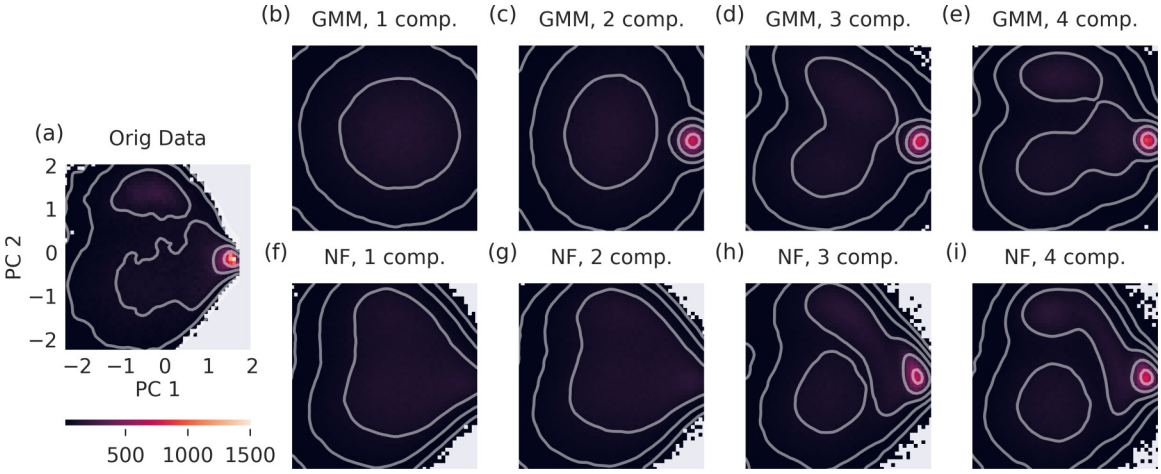


FIG. 25. Effect of number of latent components on the approximation capabilities: Marginalized distributions for the experimental data and learned models. (a) Marginalized distribution of one session of experimental data (L_RS_100817) along the first and second PCA directions. The color bar indicates the density of data points in the 2D histogram and applies to all panels. Light-gray isolines represent equal likelihood. [(b)–(e)] Marginalized probability densities for data sampled from GMMs with increasing numbers of latent components (one to four). [(f)–(i)] Corresponding distributions for NFs, also with one to four latent components.

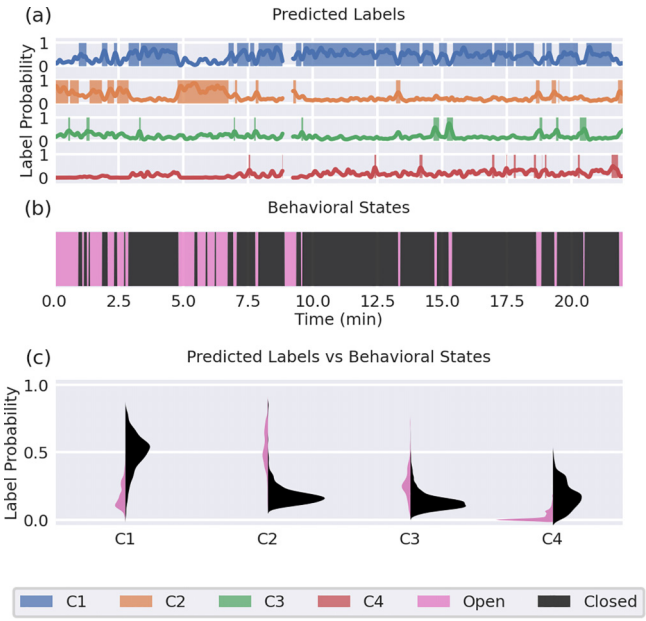


FIG. 26. Distinct mixture components in the latent space represent behavioral states (session L_RS_090817). Components are ordered by mixture weight (C1 = highest weight). (a) For each of the four components (C1–C4), smoothed (Gaussian filter, $\sigma = 3$ s), normalized likelihoods are shown as solid curves. Labels are assigned by selecting the component with the largest label probability at each time point, indicated by background shading. (b) Ground-truth behavioral states—eyes open (pink) and eyes closed (black)—are shown for direct comparison with the labels predicted in (a). (c) The label probabilities are compared against the behavioral states. For each of the components, smoothed histograms display the distributions of label probabilities for the two behavioral states (left, eyes open; right, eyes closed).

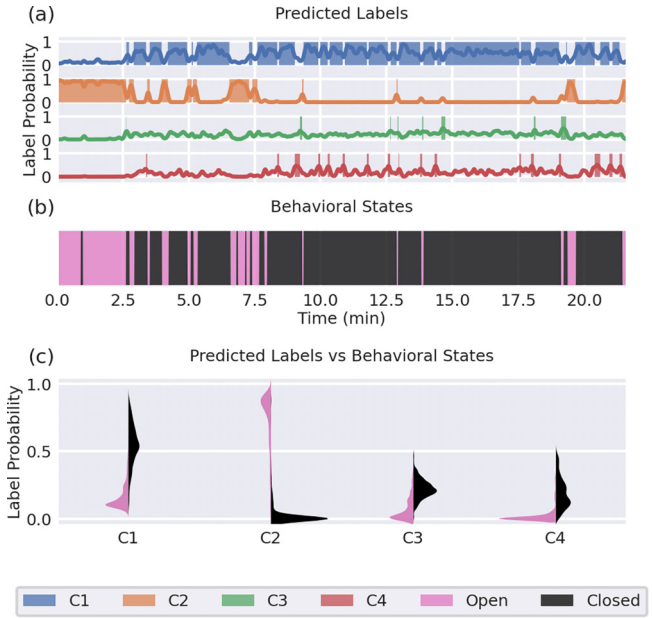


FIG. 27. Distinct mixture components in the latent space represent behavioral states (session L_RS_100817). Components are ordered by mixture weight (C1 = highest weight). (a) For each of the four components (C1–C4), smoothed (Gaussian filter, $\sigma = 3$ s), normalized likelihoods are shown as solid curves. Labels are assigned by selecting the component with the largest label probability at each time point, indicated by background shading. (b) Ground-truth behavioral states—eyes open (pink) and eyes closed (black)—are shown for direct comparison with the labels predicted in (a). (c) The label probabilities are compared against the behavioral states. For each of the components, smoothed histograms display the distributions of label probabilities for the two behavioral states (left, eyes open; right, eyes closed).

TABLE III. Scalar curvature for session (L_RS_090817) for all four components.

	Sectional curvature
Component 1	−0.020
Component 2	−0.034
Component 3	−0.048
Component 4	−0.037

TABLE IV. Scalar curvature for session (L_RS_100817) for all four components.

	Sectional curvature
Component 1	−0.024
Component 2	−0.042
Component 3	−0.035
Component 4	−0.030

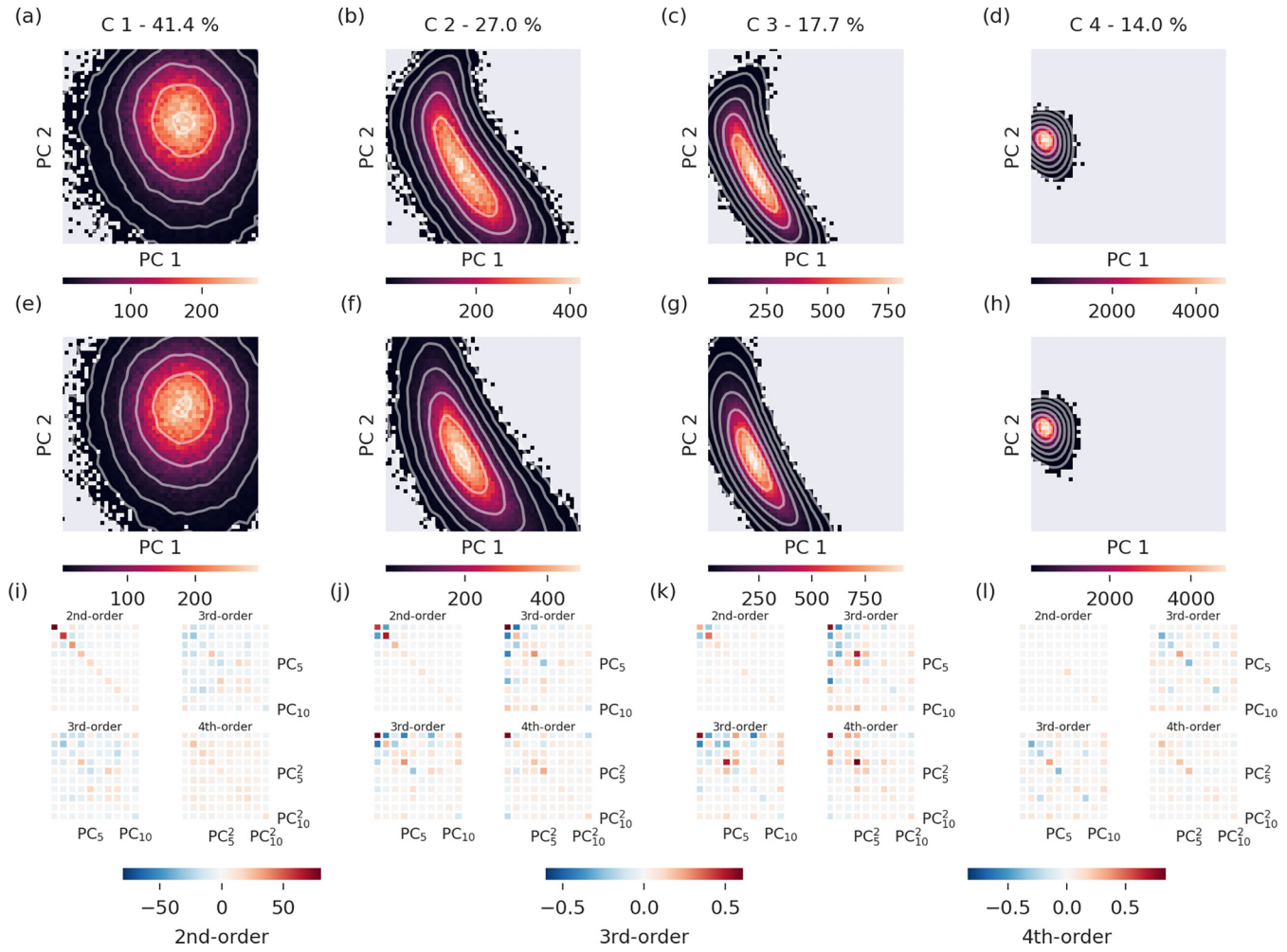


FIG. 28. Statistical characteristics of the latent components: Approximate marginalized distributions for experimental data and learned models. Components are ordered by mixture weight (C1 = highest weight). [(a)–(d)] Marginalized distributions of each latent component from the experimental session (L_RS_090817), projected onto the first two PCA directions. Light-gray contours indicate isolines of equal likelihood. The color bars indicate the density of data points in the 2D histograms. [(e)–(h)] Marginalized distribution of each component when the inverse mapping of the network is approximated by a quadratic function [Eq. (10)]. [(i)–(l)] Second- to fourth-order cumulants computed from this approximation.

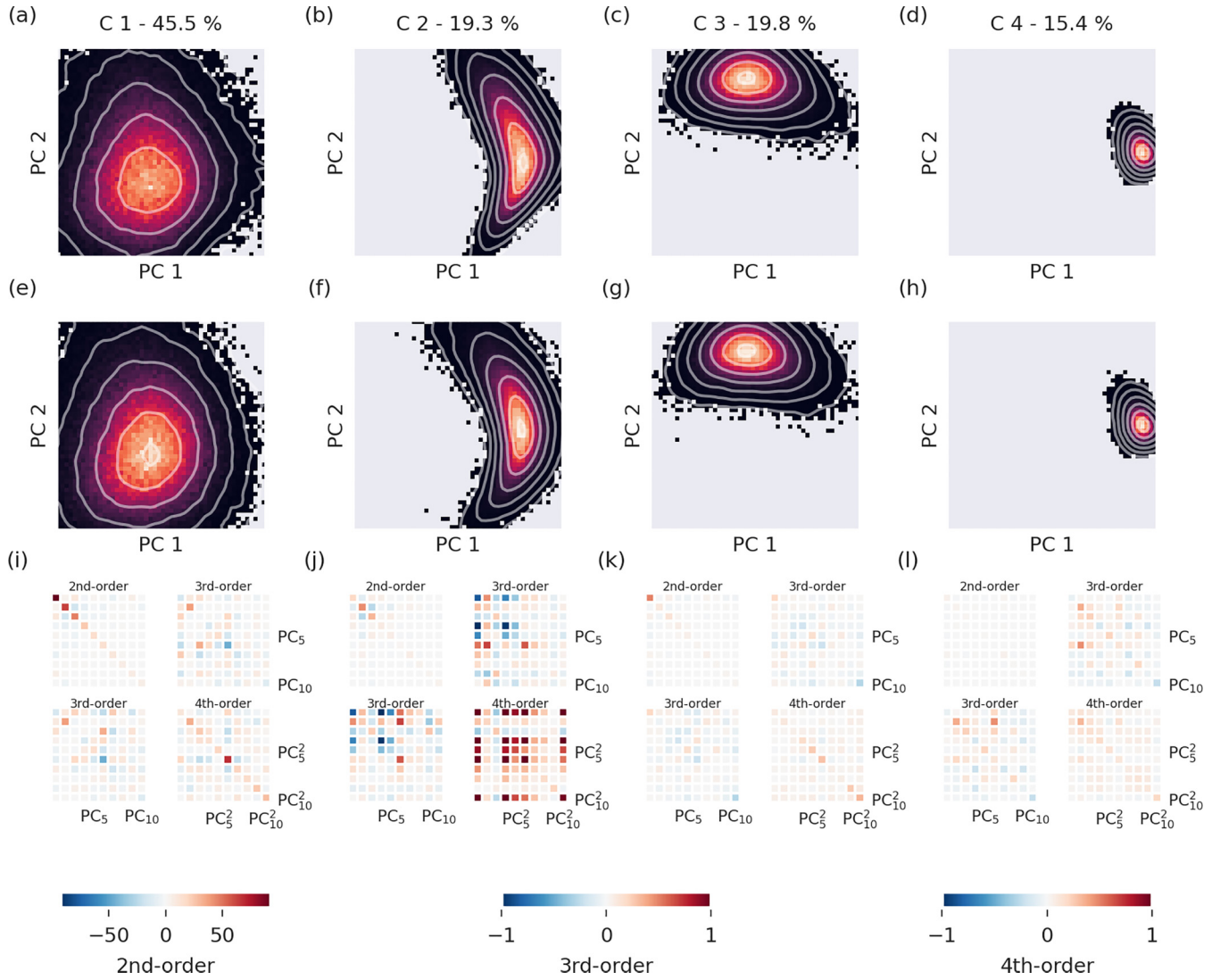


FIG. 29. Statistical characteristics of the latent components: Approximate marginalized distributions for experimental data and learned models. Components are ordered by mixture weight (C1 = highest weight). [(a)–(d)] Marginalized distributions of each latent component from the experimental session (L_RS_100817), projected onto the first two PCA directions. Light-gray contours indicate isolines of equal likelihood. The color bars indicate the density of data points in the 2D histograms. [(e)–(h)] Marginalized distribution of each component when the inverse mapping of the network is approximated by a quadratic function [Eq. (10)]. [(i)–(l)] Second- to fourth-order cumulants computed from this approximation.

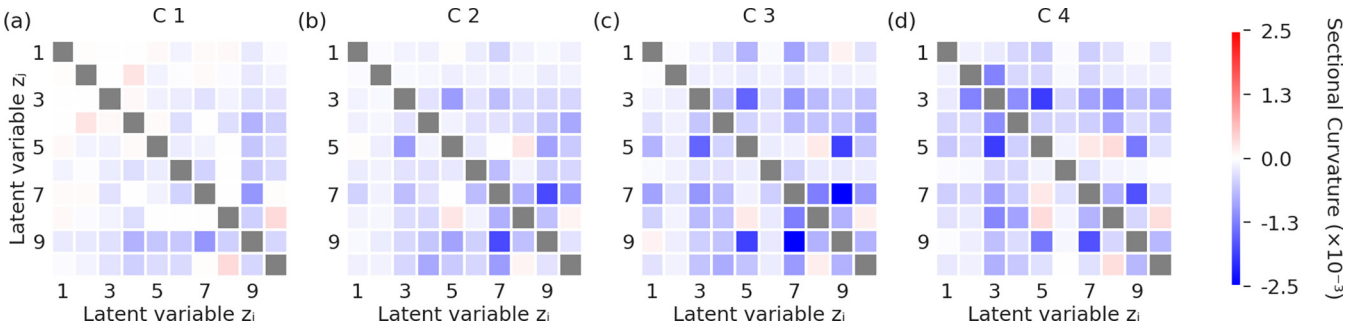


FIG. 30. Geometrical characteristics of the latent components (session L_RS_090817). Components are ordered by mixture weight (C1 = highest weight). Panels (a)–(d) show the sectional curvature for each of the four components of the latent distribution. These values are computed from the quadratically approximated inverse network. Red indicates positive curvature, while blue indicates negative curvature. The color scale is consistent across all panels, with a range defined by the values reported in Table III for the scalar curvature (i.e., the mean of sectional curvatures).

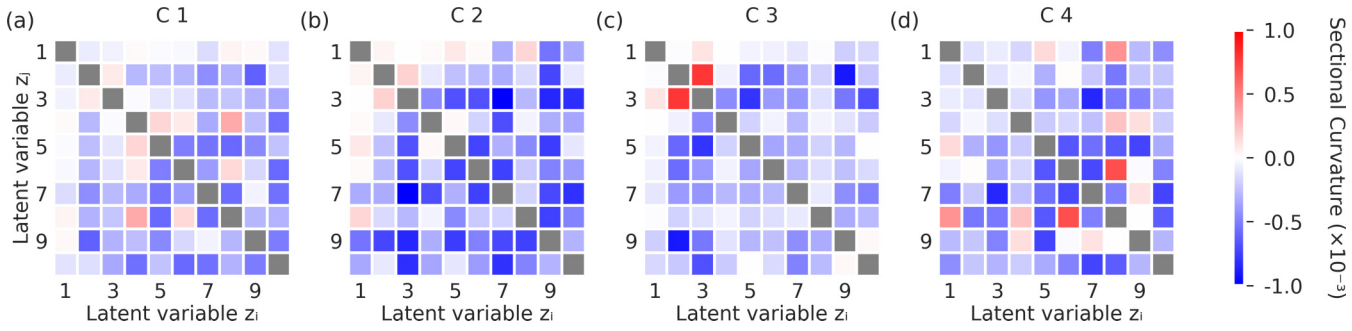


FIG. 31. Geometrical characteristics of the latent components (session L_RS_100817). Components are ordered by mixture weight (C1 = highest weight). Panels (a)–(d) show the sectional curvature for each of the four components of the latent distribution. These values are computed from the quadratically approximated inverse network. Red indicates positive curvature, while blue indicates negative curvature. The color scale is consistent across all panels, with a range defined by the values reported in Table IV for the scalar curvature (i.e., the mean of sectional curvatures).

[1] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, Neuronal population coding of movement direction, Science **233**, 1416 (1986).

[2] H. S. Seung and H. Sompolinsky, Simple models for reading neuronal population codes, Proc. Natl. Acad. Sci. USA **90**, 10749 (1993).

[3] B. A. Olshausen and D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature (London) **381**, 607 (1996).

[4] T. D. Sanger, Probability density estimation for the interpretation of neural population codes, J. Neurophysiol. **76**, 2790 (1996).

[5] W. Gerstner, A. K. Kreiter, H. Markram, and V. M. Herz, Neural codes: Firing rates and beyond, Proc. Natl. Acad. Sci. USA **94**, 12740 (1997).

[6] D. Knill and A. Pouget, The Bayesian brain: The role of uncertainty in neural coding and computation, Trends Neurosci. **27**, 712 (2004).

[7] W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget, Bayesian inference with probabilistic population codes, Nat. Neurosci. **9**, 1432 (2006).

[8] K. Doya, *Bayesian Brain: Probabilistic Approaches to Neural Coding* (MIT Press, Cambridge, UK, 2007).

[9] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, How does the brain solve visual object recognition? Neuron **73**, 415 (2012).

[10] S. Saxena and J. P. Cunningham, Towards the neural population doctrine, Curr. Opin. Neurobiol. **55**, 103 (2019).

[11] M. Jazayeri and S. Ostojic, Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity, Curr. Opin. Neurobiol. **70**, 113 (2021).

[12] S. Recanatesi, M. Farrell, G. Lajoie, S. Deneve, M. Rigotti, and E. Shea-Brown, Predictive learning as a network mechanism for extracting low-dimensional latent space representations, Nat. Commun. **12**, 1417 (2021).

[13] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy, Neural population dynamics during reaching, Nature (Lond.) **487**, 51 (2012).

[14] M. T. Kaufman, M. M. Churchland, S. I. Ryu, and K. V. Shenoy, Cortical activity in the null space: Permitting preparation without movement, Nat. Neurosci. **17**, 440 (2014).

[15] D. Sussillo, M. M. Churchland, M. T. Kaufman, and K. V. Shenoy, A neural network that finds a naturalistic solution for the production of muscle activity, Nat. Neurosci. **18**, 1025 (2015).

[16] T.-C. Kao, M. S. Sadabadi, and G. Hennequin, Optimal anticipatory control as a theory of motor preparation: A thalamo-cortical circuit model, Neuron **109**, 1567 (2021).

[17] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, Context-dependent computation by recurrent dynamics in prefrontal cortex, Nature (Lond.) **503**, 78 (2013).

[18] C. Stringer, M. Pachitariu, N. Steinmetz, M. Carandini, and K. D. Harris, High-dimensional geometry of population responses in visual cortex, Nature (Lond.) **571**, 361 (2019).

[19] T. D. Kim, T. Z. Luo, J. W. Pillow, and C. D. Brody, Inferring latent dynamics underlying neural population activity via neural differential equations, in *International Conference on Machine Learning* (PMLR, 2021), pp. 5551–5561.

[20] N. Steinemann, G. M. Stine, E. Trautmann, A. Zylberberg, D. M. Wolpert, and M. N. Shadlen, Direct observation of the neural computations underlying a single decision, eLife **12**, RP90859 (2024).

[21] C. Langdon and T. A. Engel, Latent circuit inference from heterogeneous neural responses during cognitive tasks, Nat. Neurosci. **28**, 665 (2025).

[22] C. Pandarinath, D. J. O'Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, J. M. Henderson, K. V. Shenoy, L. F. Abbott, and D. Sussillo, Inferring single-trial neural population dynamics using sequential auto-encoders, Nat. Methods **15**, 805 (2018).

[23] M. R. Whiteway and D. A. Butts, The quest for interpretable models of neural population activity, Curr. Opin. Neurobiol. **58**, 86 (2019).

[24] J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla, Neural manifolds for the control of movement, Neuron **94**, 978 (2017).

[25] J. A. Gallego, M. G. Perich, S. N. Naufel, C. Ethier, S. A. Solla, and L. E. Miller, Cortical population activity within a preserved neural manifold underlies multiple motor behaviors, Nat. Commun. **9**, 4233 (2018).

[26] J. A. Gallego, M. G. Perich, R. H. Chowdhury, S. A. Solla, and L. E. Miller, Long-term stability of cortical population dynamics underlying consistent behavior, Nat. Neurosci. **23**, 260 (2020).

[27] E. Altan, S. A. Solla, L. E. Miller, and E. J. Perreault, Estimating the dimensionality of the manifold underlying multi-electrode neural recordings, PLoS Comput. Biol. **17**, e1008591 (2021).

[28] E. Altan, X. Ma, L. E. Miller, E. J. Perreault, and S. A. Solla, Low-dimensional neural manifolds for the control of constrained and unconstrained movements, bioRxiv (2023).

[29] C. Langdon, M. Genkin, and T. A. Engel, A unifying perspective on neural manifolds and circuits for cognition, Nat. Rev. Neurosci. **24**, 363 (2023).

[30] M. Safaie, J. C. Chang, J. Park, L. E. Miller, J. T. Dudman, M. G. Perich, and J. A. Gallego, Preserved neural dynamics across animals performing similar behaviour, Nature (London) **623**, 765 (2023).

[31] C. Fortunato, J. Bennasar-Vázquez, J. Park, J. C. Chang, L. E. Miller, J. T. Dudman, M. G. Perich, and J. A. Gallego, Nonlinear manifolds underlie neural population activity during behaviour, bioRxiv (2024).

[32] A. Morales-Gregorio, A. C. Kurth, J. Ito, A. Kleinjohann, F. V. Barthélemy, T. Brochier, S. Grün, and S. J. van Albada, Neural manifolds in V1 change with top-down signals from V4 targeting the foveal region, Cell Rep. **43**, 114371 (2024).

[33] P. Gao, E. Trautmann, B. Yu, G. Santhanam, S. Ryu, K. Shenoy, and S. Ganguli, A theory of multineuronal dimensionality, dynamics and measurement, bioRxiv (2017).

[34] S. Druckmann and D. B. Chklovskii, Neuronal circuits underlying persistent representations despite time varying activity, Curr. Biol. **22**, 2095 (2012).

[35] D. Kobak, W. Brendel, C. Constantinidis, C. E. Feierstein, A. Kepecs, Z. F. Mainen, X.-L. Qi, R. Romo, N. Uchida, and C. K. Machens, Demixed principal component analysis of neural population data, eLife **5**, e10989 (2016).

[36] R. J. Gardner, E. Hermansen, M. Pachitariu, Y. Burak, N. A. Baas, B. A. Dunn, M.-B. Moser, and E. I. Moser, Toroidal topology of population activity in grid cells, Nature (London) **602**, 123 (2022).

[37] S. Schneider, J. H. Lee, and M. W. Mathis, Learnable latent embeddings for joint behavioural and neural analysis, Nature (Lond.) **617**, 360 (2023).

[38] K. Pearson, LIII. on lines and planes of closest fit to systems of points in space, Philos. Mag. **2**, 559 (1901).

[39] H. Hotelling, Analysis of a complex of statistical variables into principal components., J. Educ. Psychol. **24**, 417 (1933).

[40] R. Mitchell-Heggs, S. Prado, G. P. Gava, M. A. Go, and S. R. Schultz, Neural manifold analysis of brain circuit dynamics in health and disease, J. Comput. Neurosci. **51**, 1 (2023).

[41] C. M. Bishop, Bayesian PCA, in *Advances in Neural Information Processing Systems*, Vol. 11, edited by M. Kearns, S. Solla, and D. Cohn (MIT Press, Cambridge, UK, 1998).

[42] H. Zou, T. Hastie, and R. Tibshirani, Sparse principal component analysis, J. Comput. Graph. Stat. **15**, 265 (2006).

[43] A. H. Williams, T. H. Kim, F. Wang, S. Vyas, S. I. Ryu, K. V. Shenoy, M. Schnitzer, T. G. Kolda, and S. Ganguli, Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis, Neuron **98**, 1099 (2018).

[44] A. Pellegrino, H. Stein, and N. A. Cayco-Gajic, Dimensionality reduction beyond neural subspaces with slice tensor component analysis, Nat. Neurosci. **27**, 1199 (2024).

[45] L. Duncker and M. Sahani, Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings, Curr. Opin. Neurobiol. **70**, 163 (2021).

[46] H. Abbaspourazad, E. Erturk, B. Pesaran, and M. M. Shanechi, Dynamical flexible inference of nonlinear latent factors and structures in neural population activity, Nat. Biomed. Eng. **8**, 85 (2024).

[47] L. Dyballa, A. M. Rudzite, M. S. Hoseini, M. Thapa, M. P. Stryker, G. D. Field, and S. W. Zucker, Population encoding of stimulus features along the visual hierarchy, Proc. Natl. Acad. Sci. USA **121**, e2317773121 (2024).

[48] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, Proc. Natl. Acad. Sci. USA **102**, 7426 (2005).

[49] R. R. Coifman and S. Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. **21**, 5 (2006).

[50] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. **9**, 2579 (2008).

[51] L. McInnes, J. Healy, and J. Melville, UMAP: Uniform manifold approximation and projection for dimension reduction, J. Open Source Softw. **3**, 861 (2018).

[52] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science **290**, 2319 (2000).

[53] S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science **290**, 2323 (2000).

[54] M. Belkin and P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. **15**, 1373 (2003).

[55] M. Genkin, K. V. Shenoy, C. Chandrasekaran, and T. A. Engel, The dynamics and geometry of choice in the premotor cortex, Nature (London) **645**, 168 (2025).

[56] M. M. Monsalve-Mercado, G. M. Stine, M. N. Shadlen, and K. D. Miller, The geometry of the neural state space of decisions, bioRxiv (2025).

[57] R. Chaudhuri, B. Gerçek, B. Pandey, A. Peyrache, and I. Fiete, The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep, Nat. Neurosci. **22**, 1512 (2019).

[58] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)* (ICLR, 2014).

[59] K. Han, H. Wen, J. Shi, K.-H. Lu, Y. Zhang, D. Fu, and Z. Liu, Variational autoencoder: An unsupervised model for encoding and decoding fmri activity in visual cortex, NeuroImage **198**, 125 (2019).

[60] D. Zhou and X.-X. Wei, Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE, Adv. Neural Inf. Process. Syst. **33**, 7234 (2020).

[61] M. Schimel, T.-C. Kao, K. T. Jensen, and G. Hennequin, iLQR-VAE: Control-based learning of input-driven dynamics with applications to neural data, in *Proceedings of the 10th*

International Conference on Learning Representations (ICLR) (ICLR, 2022).

[62] T. Ahmed and L. Longo, Examining the size of the latent space of convolutional variational autoencoders trained with spectral topographic maps of eeg frequency bands, IEEE Access **10**, 107575 (2022).

[63] L. A. P. Rey, V. Menkovski, and J. W. Portegies, Diffusion variational autoencoders, in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)* (IJCAI, 2020), pp. 2704–2710.

[64] F. Koehler, V. Mehta, C. Zhou, and A. Risteski, Variational autoencoders in the presence of low-dimensional data: Landscape and implicit bias, in *Proceedings of the 10th International Conference on Learning Representations (ICLR)* (ICLR, 2022).

[65] M. Connor, G. Canal, and C. Rozell, Variational autoencoder with learned latent structure, in *International Conference on Artificial Intelligence and Statistics* (PMLR, 2021), pp. 2359–2367.

[66] G. Loaiza-Ganem, B. L. Ross, J. C. Cresswell, and A. L. Caterini, Diagnosing and fixing manifold overfitting in deep generative models, Trans. Mach. Learn. Res. (2022), https://openreview.net/forum?id=0nEZCVshxS.

[67] G. S. Alberti, J. Hertrich, M. Santacesaria, and S. Sciutto, Manifold learning by mixture models of vaes for inverse problems, J. Mach. Learn. Res. **25**, 1 (2024).

[68] M. C. Gemici, D. Rezende, and S. Mohamed, Normalizing flows on riemannian manifolds, arXiv:1611.02304.

[69] J. Brehmer and K. Cranmer, Flows for simultaneous manifold learning and density estimation, Adv. Neural Inf. Process. Syst. **33**, 442 (2020).

[70] H. Kim, H. Lee, W. H. Kang, J. Y. Lee, and N. S. Kim, Softflow: Probabilistic framework for normalizing flow on manifolds, Adv. Neural Inf. Process. Syst. **33**, 16388 (2020).

[71] A. L. Caterini, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham, Rectangular flows for manifold learning, Adv. Neural Inf. Process. Syst. **34**, 30228 (2021).

[72] C. Horvat and J.-P. Pfister, Denoising normalizing flow, Adv. Neural Inf. Process. Syst. **34**, 9099 (2021).

[73] E. Cunningham, A. D. Cobb, and S. Jha, Principal component flows, in *International Conference on Machine Learning* (PMLR, Baltimore, Maryland, 2022), pp. 4492–4519.

[74] E. Cramer, F. Rauh, A. Mitsos, R. Tempone, and M. Dahmen, Nonlinear isometric manifold learning for injective normalizing flows, arXiv:2203.03934.

[75] J. Postels, M. Danelljan, L. Van Gool, and F. Tombari, Maniflow: Implicitly representing manifolds with normalizing flows, in *2022 International Conference on 3D Vision (3DV)* (IEEE, Los Alamitos, 2022), pp. 84–93.

[76] C. Horvat and J.-P. Pfister, Intrinsic dimensionality estimation using normalizing flows, Adv. Neural Inf. Process. Syst. **35**, 12225 (2022).

[77] C. Horvat and J.-P. Pfister, Density estimation on low-dimensional manifolds: An inflation-deflation approach, J. Mach. Learn. Res. **24**, 1 (2023).

[78] K. Flouris and E. Konukoglu, Canonical normalizing flows for manifold learning, Adv. Neural Inf. Process. Syst. **36**, 27294 (2023).

[79] O. Lindenbaum, J. Stanley, G. Wolf, and S. Krishnaswamy, Geometry based data generation, Adv. Neural Inf. Process. Syst. **31**, 1400 (2018).

[80] L. Dinh, D. Krueger, and Y. Bengio, NICE: Non-linear independent components estimation, in *3rd International Conference on Learning Representations, Workshop Track* (ICLR, 2015).

[81] L. Dinh, J. Sohl-Dickstein, and S. Bengio, Density estimation using real NVP, in *Proceedings of the 5th International Conference on Learning Representations (ICLR)* (ICLR, 2017).

[82] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible 1×1 convolutions, Adv. Neural Inf. Process. Syst. **31**, 10215 (2018).

[83] A. Bekasov and I. Murray, Ordering dimensions with nested dropout normalizing flows, in ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, arXiv:2006.08777.

[84] X. Chen, A. Morales-Gregorio, J. Sprenger, A. Kleinjohann, S. Sridhar, S. J. van Albada, S. Grün, and P. R. Roelfsema, 1024-channel electrophysiological recordings in macaque V1 and V4 during resting state, Sci. Data **9**, 77 (2022).

[85] F. Draxler, S. Wahl, C. Schnörr, and U. Köthe, On the universality of volume-preserving and coupling-based normalizing flows, in *Proceedings of the 41st International Conference on Machine Learning* (PMLR, 2024), Vol. 235, pp. 11613–11641.

[86] C. Merger, A. René, K. Fischer, P. Bouss, S. Nestler, D. Dahmen, C. Honerkamp, and M. Helias, Learning interacting theories from data, Phys. Rev. X **13**, 041033 (2023).

[87] P. Izmailov, P. Kirichenko, M. Finzi, and A. G. Wilson, Semi-supervised learning with normalizing flows, in *International Conference on Machine Learning* (PMLR, 2020), pp. 4615–4630.

[88] F. Montani, R. A. Ince, R. Senatore, E. Arabzadeh, M. E. Diamond, and S. Panzeri, The impact of high-order interactions on the rate of synchronous discharge and information transmission in somatosensory cortex, Philos. Trans. R. Soc. A **367**, 3297 (2009).

[89] E. Ganmor, R. Segev, and E. Schneidman, Sparse low-order interaction network underlies a highly correlated and learnable neural population code, Proc. Natl. Acad. Sci. USA **108**, 9679 (2011).

[90] C. Gardiner, *Stochastic Methods: A Handbook for the Natural and Social Sciences*, 4th ed. (Springer, Berlin, 2009).

[91] M. Helias and D. Dahmen, *Statistical Field Theory for Neural Networks* (Springer International, Cham, 2020).

[92] J. M. Lee, *Introduction to Smooth Manifolds* (Springer, Berlin, 2012), Vol. 2.

[93] J. M. Lee, *Introduction to Riemannian Manifolds* (Springer, Berlin, 2018), Vol. 2.

[94] C. Bär, *Elementary Differential Geometry* (Cambridge University Press, Cambridge, UK, 2010).

[95] J. W. Robbin and D. A. Salamon, *Introduction to Differential Geometry* (Springer Nature, New York, 2022).

[96] E. Schneidman, M. J. Berry II, R. Segev, and W. Bialek, Weak pairwise correlations imply strongly correlated network states in a neural population, Nature (Lond.) **440**, 1007 (2006).

[97] J. H. Macke, M. Opper, and M. Bethge, Common input explains higher-order correlations and entropy in a simple model of neural population activity, Phys. Rev. Lett. **106**, 208102 (2011).

[98] S. Panzeri, M. Moroni, H. Safaai, and C. D. Harvey, The structures and functions of correlations in neural population codes, Nat. Rev. Neurosci. **23**, 551 (2022).

[99] I. Jones, Manifold diffusion geometry: Curvature, tangent spaces, and dimension, arXiv:2411.04100.

[100] H. Shao, A. Kumar, and P. Thomas Fletcher, The riemannian geometry of deep generative models, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (IEEE, Los Alamitos, CA, 2018), pp. 315–323.

[101] F. Acosta, S. Sanborn, K. D. Duc, M. Madhav, and N. Miolane, Quantifying extrinsic curvature in neural manifolds, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, Los Alamitos, CA, 2023), pp. 610–619.

[102] P. T. Fletcher, S. Venkatasubramanian, and S. Joshi, The geometric median on riemannian manifolds with application to robust atlas estimation, NeuroImage **45**, S143 (2009).

[103] A. C. Gilbert and K. O'Neill, CA-PCA: Manifold dimension estimation, adapted for curvature, SIAM J. Math. Data Sci. **7**, 355 (2025).

[104] H. Farooq, Y. Chen, T. T. Georgiou, A. Tannenbaum, and C. Lenglet, Network curvature as a hallmark of brain structural connectivity, Nat. Commun. **10**, 4937 (2019).

[105] M. Weber, J. Stelzer, E. Saucan, A. Naitsat, G. Lohmann, and J. Jost, Curvature-based methods for brain network analysis, arXiv:1707.00180.

[106] R. Azeredo da Silveira and F. Rieke, The geometry of information coding in correlated neural populations, Annu. Rev. Neurosci. **44**, 403 (2021).

[107] R. Moreno-Bote, J. Beck, I. Kanitscheider, X. Pitkow, P. Latham, and A. Pouget, Information-limiting correlations, Nat. Neurosci. **17**, 1410 (2014).

[108] A. Kohn, R. Coen-Cagli, I. Kanitscheider, and A. Pouget, Correlations and neuronal population information, Annu. Rev. Neurosci. **39**, 237 (2016).

[109] A. Pouget, P. Dayan, and R. S. Zemel, Inference and computation with population codes, Annu. Rev. Neurosci. **26**, 381 (2003).

[110] S. Chung and L. F. Abbott, Neural population geometry: An approach for understanding biological and artificial neural networks, Curr. Opin. Neurobiol. **70**, 137 (2021).

[111] C.-N. Chou, R. Kim, L. A. Arend, Y.-Y. Yang, B. D. Mensh, W. M. Shim, M. G. Perich, and S. Chung, Geometry linked to untangling efficiency reveals structure and computation in neural populations, bioRxiv (2025).

[112] A. E. Urai, B. Doiron, A. M. Leifer, and A. K. Churchland, Large-scale neural recordings call for new insights to link brain and behavior, Nat. Neurosci. **25**, 11 (2022).

[113] E. Cramer, A. Mitsos, R. Tempone, and M. Dahmen, Principal component density estimation for scenario generation using normalizing flows, Data-Cent. Eng. **3**, e7 (2022).

[114] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen, Understanding and mitigating exploding inverses in invertible neural networks, in *International Conference on Artificial Intelligence and Statistics* (PMLR, 2021), pp. 1792–1800.

[115] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, beta-VAE: Learning basic visual concepts with a constrained variational framework, in *International Conference on Learning Representations* (ICLR, 2017).

[116] W.-N. Hsu, Y. Zhang, and J. Glass, Unsupervised learning of disentangled and interpretable representations from sequential data, Adv. Neural Inf. Process. Syst. **30**, 1878 (2017).

[117] H. Kim and A. Mnih, Disentangling by factorising, in *International Conference on Machine Learning* (PMLR, 2018), pp. 2649–2658.

[118] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, Isolating sources of disentanglement in variational autoencoders, Adv. Neural Inf. Process. Syst. **31**, 2610 (2018).

[119] M. H. Sarhan, A. Eslami, N. Navab, and S. Albarqouni, Learning interpretable disentangled representations using adversarial vaes, in *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data: First MICCAI Workshop, DART 2019, and First International Workshop, MIL3ID 2019, Shenzhen, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13 and 17, 2019, Proceedings 1* (Springer, Berlin, 2019), pp. 37–44.

[120] X. Zhu, C. Xu, and D. Tao, Learning disentangled representations with latent variation predictability, in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16* (Springer, Berlin, 2020), pp. 684–700.

[121] X. Zhu, C. Xu, and D. Tao, Where and what? Examining interpretable disentangled representations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, Los Alamitos, CA, 2021), pp. 5861–5870.

[122] M. M. Churchland, B. M. Yu, J. P. Cunningham, L. P. Sugrue, M. R. Cohen, G. S. Corrado, W. T. Newsome, A. M. Clark, P. Hosseini, B. B. Scott, D. C. Bradley, M. A. Smith, A. Kohn, J. A. Movshon, K. M. Armstrong, T. Moore, S. W. Chang, L. H. Snyder, S. G. Lisberger, N. J. Priebe, *et al.*, Stimulus onset quenches neural variability: A widespread cortical phenomenon, Nat. Neurosci. **13**, 369 (2010).

[123] R. R. de Ruyter van Steveninck, G. D. Lewen, S. P. Strong, R. Koberle, and W. Bialek, Reproducibility and variability in neural spike trains, Science **275**, 1805 (1997).

[124] G. L. Gerstein and B. Mandelbrot, Random walk models for the spike activity of a single neuron, Biomed. Pharmacol. J. **4**, 41 (1964).

[125] M. A. Buice, J. D. Cowan, and C. C. Chow, Systematic fluctuation expansion for neural network activity equations, Neural Comput. **22**, 377 (2010).

[126] P. Bouss, S. Nestler, K. Fischer, C. Merger, L. Schutzeichel, S. Musall, A. René, and M. Helias, Code repository for the paper "Characterizing neural manifolds' properties and curvatures using normalizing flows" (PRX Life, 2026), Zenodo (2026), doi:10.5281/zenodo.19591898.

[127] N. A. Steinmetz, C. Aydin, A. Lebedeva, M. Okun, M. Pachitariu, M. Bauza, M. Beau, J. Bhagat, C. Böhm, M. Broux, S. Chen, J. Colonell, R. J. Gardner, B. Karsh, F. Kloosterman, D. Kostadinov, C. Mora-Lopez, J. O'Callaghan, J. Park, J. Putzeys, *et al.*, Neuropixels 2.0: A miniaturized high-density

probe for stable, long-term brain recordings, Science **372**, eabf4588 (2021).

[128] E. Balla, G. Nabbefeld, C. Wiesbrock, J. Linde, S. Graff, S. Musall, and B. M. Kampa, Broadband visual stimuli improve neuronal representation and sensory perception, Nat. Commun. **16**, 2957 (2025).

[129] M. Pachitariu, S. Sridhar, J. Pennington, and C. Stringer, Spike sorting with Kilosort4, Nat. Methods **21**, 914 (2024).

[130] A. Montanari and E. Richard, A statistical model for tensor PCA, Adv. Neural Inf. Process. Syst. **27**, 2897 (2014).

[131] G. Ben Arous, C. Gerbelot, and V. Piccolo, Langevin dynamics for high-dimensional optimization: The case of multi-spiked tensor PCA, arXiv:2408.06401.

[132] G. Ben Arous, C. Gerbelot, and V. Piccolo, Stochastic gradient descent in high dimensions for multi-spiked tensor PCA, arXiv:2410.18162.